Berufsakademie Sachsen
Staatliche Studienakademie Dresden
Studienrichtung Informationstechnik

Sächsische Landesbibliothek –
Staats- und Universitätsbibliothek
Dresden

Entwurf und Realisierung einer Configuration Management Database nach ITIL für den Service Desk der SLUB Dresden

Diplomarbeit zur Erlangung des Grades Diplom-Ingenieur (BA)
in der Studienrichtung Informationstechnik

eingereicht von:

Frank Niebling
03.02.1982

1. Gutachter: Herr Dr.-Ing. Lutz Kowalke
2. Gutachter: Herr Dipl.-Math. Rico Barth

Tag der Themenübergabe: 30.04.2009
Tag der Einreichung: 30.07.2009

Autorenreferat

NIEBLING, Frank: Entwurf und Realisierung einer Configuration Management Database nach ITIL für den Service Desk der SLUB Dresden, Berufsakademie Sachsen, Staatliche Studienakademie Dresden, Informationstechnik, Diplomarbeit, 2009.
64 Seiten, 9 Literaturquellen, 17 Onlinequellen, 9 Anlagen

In der Arbeit wird der Entwurf und die Realisierung einer Configuration Management Database für den Service Desk der SLUB Dresden betrachtet. Es wird auf Grundlage von ITIL und den Anforderungen der SLUB ein Konzept zum Aufbau und zur Strukturierung der dabei relevanten Daten vorgestellt. Für den Import der Daten wird ein automatisiertes System geschaffen, das Daten aus verschiedenen Quellen mit der Configuration Management Database in regelmäßigen Abständen abgleicht. Der Service Desk erhält dadurch einen zentralen Zugriff auf Daten zu physikalischen Geräten sowie virtuellen Maschinen und kann diese in die Arbeitsprozesse integrieren. Zusätzlich wird das System durch den scriptbasierten Import von Verlinkungen zwischen virtuellen und physikalischen Servern und Ergebnissen aus der Geräte- und Netzwerküberwachung ergänzt. Schließlich wird betrachtet, welche weiteren Entwicklungsschritte auf Grundlage der vorgenommenen Änderungen möglich worden und welche Aspekte bei der Umsetzung dieser Schritte eine Rolle spielen.

Inhalt

Verzeichnis der verwendeten Abkürzungen

BVZ	Bestandsverzeichnis
CI	Configuration Item
CMDB	Configuration Management Database
CMS	Configuration Management System
CSV	Comma-Separated Values
DB	Datenbank
DNS	Domain Name System
FAQ	Frequently Asked Questions
KB	Kilobyte
KEDB	Known Error Database
ID	Identifikator
IT	Informationstechnik
ITIL	IT Infrastructure Library
ITSM	IT Service Management
MB	Megabyte
NRPE	Nagios Remote Plugin Executor
OTRS	Open Ticket Request System
SKMS	Service Knowledge Management System
SLUB	Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden
SQL	Structured Query Language
URL	Uniform Resource Locator
UTF-8	8-bit Unicode Transformation Format
VM	Virtual Machine
XML	Extended Markup Language

1 Motivation und Ziele der Arbeit

Der *Service Desk* hat sich in vielen Unternehmen als Kommunikationsschnittstelle zwischen der IT-Abteilung und den Anwendern im internen und externen Umfeld etabliert. Auch an der SLUB Dresden werden seit einigen Jahren gezielt Mitarbeiter und Werkzeuge eingesetzt, um diese Schnittstelle bereitzustellen. Zwar stecken an einer öffentlichen Bibliothek kaum kommerziellen Interessen hinter dem Betrieb des Service Desks, aber sein Einsatz ist trotzdem von Bedeutung. Zum einen hilft er die Vorgänge im Support effektiver zu gestalten und damit Arbeitszeit und Kosten zu sparen. Zum anderen dient er dazu, die allgemeine Zufriedenheit in Bezug auf die Arbeit der IT-Abteilung zu verbessern. Dies gelingt dadurch, dass er den Anwendern die Möglichkeit gibt, ihre Anliegen gezielt weiterzureichen und daraufhin schnelle, kompetente Hilfe zu erhalten. Die grundlegenden Anforderungen, die gegenüber dem Service Desk im Allgemeinen bestehen, wurden bereits umgesetzt. Mit Unterstützung verschiedener Werkzeuge werden die Meldungen der Anwender erfasst, analysiert und abgearbeitet. Im Rahmen dieser Vorgänge sind aber durchaus noch Verbesserungen möglich. Insbesondere aus dem Bereich der *IT Infrastructure Library* (ITIL) lassen sich Ansätze dafür finden. Einer dieser Ansätze ist die Anbindung der vorhandenen Werkzeuge an eine Datenbank mit technischen Gerätedaten. Die theoretische und praktische Umsetzung der Anbindung wird ein wesentlicher Bestandteil dieser Arbeit sein. Aber auch die Strukturierung der Gerätedaten sowie die Integration von Ergebnissen aus der Netzwerküberwachung wird eine Rolle spielen. Hauptziel ist dabei stets, eine praktische Verbesserung für die Arbeit des Service Desks an der SLUB zu erzielen beziehungsweise die technische Basis dafür zu schaffen.

Im folgenden Abschnitt wird zuerst eine Analyse des aktuellen Zustands im Bezug auf den Einsatz des Service Desks an der SLUB erfolgen. Es soll dargelegt werden, welche Anforderungen aus Sicht der zuständigen Mitarbeiter bestehen und wie die Empfehlungen aus ITIL berücksichtigt werden können. Daraus werden mögliche Verbesserungsschritte abgeleitet und der Lösungsansatz sowie die Ziele für einen ausgewählten Schritt näher beschrieben.

Im dritten Abschnitt wird ein konkretes Konzept zur Umsetzung einer *Configuration Management Database* (CMDB) und deren Strukturierung vorgestellt. In diesem Zusammenhang wird auch ein System entstehen, dass für den automatischen

Abgleich bestehender Daten aus mehreren Systemen und die Einbindung der Netzwerküberwachung sorgt.
Welche Bedeutung die Änderungen für die Arbeit der Mitarbeiter des Service Desks und die Administratoren haben, wird im vierten Abschnitt beschrieben. Dabei wird sowohl der Nutzen der hinzugekommenen Daten, als auch der zusätzliche Pflegeaufwand betrachtet.
Schließlich wird analysiert, ob die vorgeschlagenen Maßnahmen die gestellten Ziele und Anforderungen erfüllen. Zusätzlich erfolgt eine kurze Betrachtung von weiteren Entwicklungsmöglichkeiten, die sich aus der Arbeit ergeben haben, aber noch nicht umgesetzt werden konnten.

2 Zustandsanalyse und Schritte zur Verbesserung

2.1 Grundlagen und aktueller Zustand

An der SLUB Dresden bildet der Service Desk keine separate Organisationseinheit, sondern wird im Rahmen der täglichen Aufgaben von den Mitarbeitern der IT-Abteilung betrieben. Als Schnittstelle zu den Anwendern dient primär die *Hotline*, ein Schnurlostelefon, das sich beim mit dem Hotlinedienst beauftragten Mitarbeiter befindet. Es existieren aber auch E-Mail-Adressen, über die verschiedenste Anliegen der Anwender aufgenommen werden.
Als zentrales Werkzeug zur Erfassung und Verwaltung aller Kundenmeldungen und der daraus resultierenden Maßnahmen dient das Open Ticket Request System (OTRS). Ein weiterer Einsatzzweck des OTRS ist die Bearbeitung von verschiedenen Verwaltungsaufgaben, wie Beschaffungs- oder Dienstleistungsanträgen. Für die Abarbeitung aller Meldungen wurden Workflows geschaffen und den Mitarbeitern der Abteilung entsprechende Rechte als Agenten auf dem System eingeräumt.
Neben dem OTRS existiert noch ein System aus Eigenentwicklung für die Verwaltung der Daten aller technischen Geräte im IT-Bereich - das Bestandsverzeichnis (BVZ). Dieses System wird auch dazu genutzt sämtliche Störungen sowie Reparatur- und Wartungsmaßnahmen, die ein Gerät betreffen, zu dokumentieren.

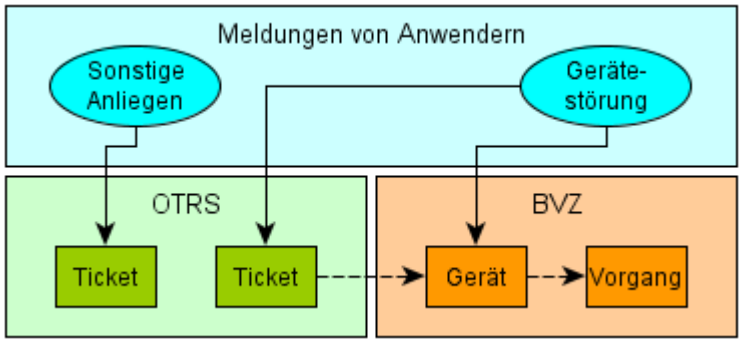

Abbildung 1: Ablauf der Bearbeitung von Anwendermeldungen

Wie in Abbildung 1 zu erkennen ist, ergeben sich verschiedene Wege wie Meldungen von Anwendern bearbeitet werden. Anliegen, die allgemeine Anfragen oder Probleme mit dem Serviceangebot der IT betreffen, werden im OTRS erfasst und dort bearbeitet. Störungen von Geräten können ebenfalls im OTRS erfasst werden. Allerdings führt dies dazu, dass die zuständigen Mitarbeiter, parallel zur Bearbeitung im OTRS, einen Vorgang für das Gerät im BVZ erstellen, bei dem Zustand und durchgeführte Maßnahmen dokumentiert werden. Alternativ besteht die Möglichkeit die Störung direkt an das BVZ zu melden, wodurch für den gesamten Vorfall meistens keine Dokumentation im OTRS zustande kommt.
Für die Bearbeitung der Meldungen wird im OTRS ein Ticket angelegt. Die Erfassung eines Tickets erfolgt nach einem festen Schema, das im Folgenden kurz aufgeführt ist:

- Erfassen der Kundendaten
- Erfassen aller relevanten Informationen zum Anliegen des Kunden
- Einordnung nach Art der Meldung (Typ)
- Einordnung des Tickets in einen bestimmten Zuständigkeitsbereich (Queue)

Wenn das Problem bzw. Anliegen bereits bei der Annahme geklärt werden konnte, wird der Vorgang direkt abgeschlossen. Ansonsten geht die Bearbeitung an die zuständigen Mitarbeiter über (funktionale Eskalation). Dabei werden zusätzliche Informationen, die sich bei der Bearbeitung ergeben, dem Ticket als Artikel hinzugefügt. Nach Abschluss der Bearbeitung wird das Ticket geschlossen. Wenn es sich um die Bearbeitung einer Störung oder eines Problems handelt, wird dem Ticket für die statistische Auswertung eine Fehlerkategorie zugeordnet.

2.2 Anforderungen und Problemstellung

2.2.1 Der Service Desk in ITIL

Bei vielen Vorgängen, die an den unterschiedlichsten Orten auf ähnliche Weise durchgeführt werden müssen, setzten sich irgendwann bestimmte Vorgehensweisen durch. Auch für den IT-Bereich gibt es solche De-facto-Standards. Einer davon ist die IT Infrastructure Library.
ITIL wird auf der offiziellen Website als einzige einheitliche und umfassende Dokumentation von *Best Practices* für das IT Service Management (ITSM) beschrieben [a]. Unter Best Practices versteht man dabei „Aktivitäten oder Prozesse, deren Einsatz in mehreren Organisationen nachweislich zum gewünschten Erfolg geführt hat“ [b]. Im Kern beschreibt ITIL in fünf Publikationen, was bei der Entwicklung, Einführung und Pflege eines Services (*Service Lifecycle*) zu beachten ist und gibt Vorschläge zu verschiedenen Bereichen des Service Management, die je nach den individuellen Anforderungen umgesetzt werden können. ITIL dient auch dazu, eine einheitliche Begriffsstruktur für das Gebiet zu schaffen. In den folgenden Abschnitten dieser Arbeit werden diese Begriffe verstärkt verwendet. Die Bedeutung lässt sich bei Bedarf im *ITIL V3 Glossar* des ITSM Forum nachschlagen [b].

Der Service Desk ist eine Funktion, die im Bereich der Service Operation, also dem vierten Buch von ITIL, beschrieben wird [1]. Er dient als *Single Point of Contact* zwischen Anwendern und dem Provider eines Services, an dem alle *Incidents* und *Service Requests* bearbeitet werden. Das Ziel des Service Desks ist die Wiederherstellung des normalen Service-Betriebs. Durch die Bündelung der Ressourcen hilft er, die Qualität sowie die Geschwindigkeit bei der Bearbeitung von Anwenderanliegen zu verbessern.
In ITIL wird viel über den Aufbau bzw. den Einsatz von Personal am Service Desk geschrieben. Auch die hierarchische Gliederung des Service Desks in First-, Second- und Third-Level-Support spielt eine Rolle. Diese Aspekte treten hier in den Hintergrund, da in dieser Hinsicht an der SLUB wenig Potential für Änderungen besteht. Es wird vorrangig betrachtet, welche Werkzeuge und Informationsquellen laut ITIL für das Personal des Service Desks hilfreich sind und wie diese eingesetzt werden.

Der Servicedesk sollte Zugang zu folgenden Informationen haben[1] [2]:

- Aufzeichnungen und Informationen zu allen Incidents
- Aufzeichnungen und Informationen zu Problemen
- Known Error Database
- Change Schedule
- Quellen mit internem Wissen (insb. Experten)
- Service Knowledge Management System
- Configuration Management System
- Alarmmeldungen aus Überwachungssoftware

Obwohl der Service Desk an der SLUB nicht bewusst nach den Konzepten von ITIL aufgebaut wurde, sind die aufgeführten Informationen weitestgehend zugänglich. In Tabelle 1 wird aufgezeigt, in welcher Form diese vorliegen.

Information laut ITIL	**Quelle an der SLUB**
Aufzeichnung von Incidents	Tickets im OTRS, Vorgangsdokumentation im BVZ
Aufzeichnung von Problemen	Tickets und FAQ-Einträge im OTRS
Quellen mit internem Wissen	Einträge im internen Wiki, zuständige Mitarbeiter
Service Knowledge Management System	Einträge im internen Wiki, BVZ, OTRS
Configuration Management System	BVZ
Known Error Database	Tickets, FAQ-Einträge im OTRS
Meldungen aus Überwachungs-software	Zugriff auf Nagios (Mailbenachrichtigung, Webinterface, Firefox-Plugin), Syslog-Meldungen (Mail)

Tabelle 1: Verfügbarkeit von Informationsquellen für den Service Desk

[1] In der Quelle [2] werden diese Informationen als wichtig für einen *outsourced Service Desk beschrieben*. Es gilt aber, dass jede Form des Service Desks einen Zugang zu diesen Informationen haben sollte.

Ein Bereich, der aufgrund seiner Komplexität näher betrachtet werden soll, ist der Aufbau des Systems für das Service Knowledge Management. Das SKMS ist eine „Sammlung von Hilfsmitteln und Datenbanken, die zur Verwaltung von Wissen und Informationen verwendet werden“ [b]. Neben dem CMS und der KEDB beinhaltet es alles, was zur Verwaltung von Services relevant ist. Beim CMS handelt es sich um einen „Satz an Hilfsmitteln und Datenbanken, der für die Verwaltung der Configuration-Daten […] verwendet wird“ [b]. Neben der reinen Verwaltung von Komponentendaten in der CMDB umfasst das System auch Informationen zu Incidents und allen weiteren Daten, die in Bezug zu den Komponenten stehen können [3]. Wichtig ist, dass das CMS verschiedene Sichten für verschiedene Interessengruppen anbietet. So benötigt zum Beispiel der Service Desk andere Informationen als das Qualitätsmanagement.
Bei der Erstellung der CMDB kann das CMS auf mehrere vorhandene Teil-CMDBs zugreifen und diese zu einer gemeinsamen zusammenführen [3]. Soweit es möglich ist, sollte das Einpflegen und Aktualisieren der Daten in der CMDB durch automatische Prozesse erfolgen, um die Kosten zu reduzieren und Fehler zu vermeiden. Schnittstellen zwischen CMS und verschiedenen Tools, die diese Prozesse unterstützen, können verwendet werden. Sie sollen nicht nur dabei helfen neue Daten einzuspielen, sondern ermöglichen auch Vergleiche zwischen dem aktuellen Zustand und den Daten im CMS.

Wesentliches Element in der CMDB ist das Configuration Item (CI) [4]. Es handelt sich dabei um eine Ressource, eine Service-Komponente oder einen anderen Gegenstand, der sich unter der Kontrolle des Configuration Managements befindet. Was man alles zu den CIs zählt, hängt davon ab, wie detailliert deren Daten gebraucht werden. So kann zum Beispiel ein Computer als Configuration Item gesehen werden. Ob man dessen Hard- und Softwarekomponenten auch als CI erfasst, muss je nach Bedarf festgelegt werden. CIs mit ähnlichen Eigenschaften werden zur besseren Übersichtlichkeit gruppiert und klassifiziert. Die Identifikation eines CIs muss nicht nur in der CMDB problemlos möglich sein, sondern sollte sich auch soweit möglich auf dem physischen Objekt widerspiegeln, indem man dieses mit einer entsprechenden Markierung versieht [5]. Welche Attribute man den CIs hinzufügt, hängt von der gewählten CI-Struktur und dem Bedarf nach detaillierten Informationen ab.

Neben dem Identifikationsmerkmal zählen folgende Werte zu den typischen Attributen [5]:

- Name
- Version
- Standort
- Auslieferungsdatum
- Lizenzdaten
- Besitzer
- Status
- Hersteller

Zusätzlich zählen auch alle Informationen, die die Beziehungen zwischen einzelnen Configuration Items darstellen, zu den gespeicherten Daten.

In der KEDB werden Aufzeichnungen zu allen bekannten Vorfällen und Fehlern gespeichert [6]. Diese umfassen nicht nur Details zu den Symptomen, sondern auch Beschreibungen für Workarounds oder Lösungen, die dabei helfen die Funktion eines Services wiederherzustellen. Bei wiederkehrenden Vorfällen können dadurch schneller Diagnosen und Lösungen gefunden werden. Das Personal des Service Desks muss dafür mit den Funktionen und dem Inhalt der KEDB vertraut gemacht werden. Es muss außerdem sicher gestellt werden, dass neue Einträge in der KEDB formalen Kriterien entsprechen und doppelte Einträge vermieden werden. Besonders vorteilhaft ist es ein Tool zu verwenden, das möglichst einfach aus den Aufzeichnungen zu einem Incident einen Eintrag in der KEDB erzeugt.

Die Bereitstellung all dieser Komponenten ist Aufgabe des Service Asset and Configuration Management (SACM) [7]. In erster Linie sollen durch dieses die anderen Prozesse des Service Management unterstützt werden. Zu den Aufgaben des SACM zählen unter anderem die Verwaltung und Verantwortung für Configuration Items und Service Assets in allen Phasen des Service Lifecycle. Dazu gehört auch, die Autorisierung von Änderungen und die Bewahrung der Integrität aller Daten für ein korrekt gepflegtes CMS. Ein weiteres Ziel des Configuration Managements ist die Erstellung eines *Configuration Model* [8]. Dieses stellt anhand der CIs die Beziehungen zwischen allen Services, Ressourcen und der Infrastruktur

dar. Das Model hilft unter anderem bei der Einschätzung von Auswirkung und Ursache von Incidents und Problemen, aber auch bei der Planung von Änderungen und Weiterentwicklungen. Das Model dient zudem als Richtlinie an dem sich das IT Service Management bei allen Maßnahmen orientiert. Wie umfassend die dafür notwendigen Informationen angelegt werden, wird davon abhängig gemacht, wie sehr diese benötigt werden [9]. Wichtig ist, dass für jeden Service deutlich gemacht wird, auf welchen Komponenten er basiert.

2.2.2 Anforderungen der SLUB und Probleme mit dem aktuellen Zustand

Neben den Vorschlägen von ITIL gibt es auch Anforderungen aus der SLUB, insbesondere auch aus dem Bereich der Mitarbeiter in der IT-Abteilung, gegenüber dem Service Desk. Wie schon in der Einführung erwähnt, wurden die grundlegenden Anforderungen bereits umgesetzt und sollen daher hier keine weitere Beachtung finden. Vielmehr wird es darum gehen was noch erreicht werden soll.
Ein Aspekt ist die zukünftige Nutzung des BVZ. Dieses erfüllt zwar momentan seinen Zweck, ist aber langfristig gesehen nur unter hohem Aufwand erweiterbar. Das liegt vor allem daran, dass es kein verbreitetes System ist, sondern eine Eigenentwicklung mit dessen technischen Grundlagen sich nur wenige Personen auskennen. Größere Erweiterungen können nur durch Anpassungen in den Perl-Scripten und dem komplexen Datenbankschema gelingen, was einen umfangreichen Entwicklungsaufwand bedeuten würde. Daher ist es auch nicht möglich das BVZ mit einem Trouble Ticket System zu erweitern, das über die Erfassung von Gerätestörungen hinausgeht. Der Einsatz eines Alternativsystems wie OTRS ist unvermeidbar. Allerdings ist es kurzfristig auch nicht möglich die Funktionen des BVZ vollständig durch OTRS zu ersetzten. Dies liegt vor allem daran, dass die Bestandsdaten im BVZ von haushaltsrechtlicher Bedeutung sind. Dass das BVZ die in dieser Hinsicht bestehenden Anforderungen erfüllt, wurde geprüft und das System darauf für den Bestandsnachweis zugelassen.
Diese Nutzung von zwei verschiedenen Systemen wirkt sich wie in Abbildung 1 zu sehen ist, letztendlich auf den Service Desk aus. Gerätestörungen werden umständlich in zwei parallelen Vorgängen bearbeitet oder sind den Mitarbeitern im First-Level-Support überhaupt nicht ersichtlich. Ein Zugriff auf die Dokumentation der vorgenommenen Maßnahmen gestaltet sich für den First-Level-Support als ebenso

schwierig. Perspektivisch ist es daher von Vorteil, wenn die Gerätestörungen im gleichen System behandelt werden wie alle anderen Incidents. Voraussetzung ist dabei, dass die Gerätedaten, die im BVZ zur Verfügung stehen, auch in diesem System vorhanden sind.

In diesem Zusammenhang muss auch der Aspekt der Verfügbarkeit von weitergehenden Informationen für die Behebung von Incidents betrachtet werden. Der Zugriff auf Gerätedaten ist nur über den Umweg BVZ möglich. Sofern man dort keine erweiterten Rechte hat, ist die Arbeit mit dem System durchaus umständlich, da man zum Auffinden eines Gerätes nur nach Inventar- und Seriennummern suchen kann. Zusätzlich ist auch das Auffinden relevanter Dokumentationen im internen Wiki schwierig. Dort angelegte Einträge sind auch nicht mit Daten im BVZ oder OTRS verknüpft. Für weiterführende Informationen zu einem Incident muss man erst den passenden Eintrag im Wiki suchen, wobei es oftmals nicht sichergestellt ist, dass man die gewünschten Informationen dort auch findet bzw. dass diese überhaupt existieren. Eine zentrale Sammlung diese Informationen wäre daher hilfreich und würde vor allem die Informationsfindung beim wiederholten Auftreten von Incidents verbessern.

Im Gegensatz zu allen physikalischen Servern, die wie normale Computer im BVZ erfasst werden, sind virtuelle Server dort nicht zu finden. Allerdings spielen sie für die Arbeit am Service Desk eine genau so wichtige Rolle. Ihre Daten müssen daher auch für den unmittelbaren Zugriff zur Verfügung stehen. An der SLUB laufen rund 200 virtuelle Maschinen auf 15 Servern mit VMware ESX als Basis. Sie dienen vorwiegend als technische Plattform für verschiedene Verfahren. Auch der OTRS-Server läuft auf einer dieser VMs. Wenn es zu einer Störung auf einer der VMs kommt, lassen sind ohne weiterführende Zugriffsrechte nur zwei Informationsquellen für den Support nutzen. Dies sind zum einen die Daten, die vom *Virtual Center* (das Managementsystem für VMs) geliefert werden und zum anderen die Ergebnisse der mit Nagios durchgeführten Netzwerk- und Geräteüberwachung.

Abgesehen von dem Zusammenspiel mit anderen Tools, gibt es auch konkrete Anforderung in Bezug auf OTRS allein. Die aktuell im Einsatz befindliche Version 2.2.7 wird durch die Version 2.3.4 ersetzt werden. Primär dient die Umstellung zur Schaffung einer Basis für die Implementation neuer Funktionen ins OTRS. Vor allem im Bereich der Bearbeitung von Beschaffungsanträgen, ist mittelfristig eine Erweiterung geplant. Durch die neue Version wird es aber auch möglich zwei

Zusatzmodule im OTRS zu vereinen, die sich in der alten Version noch gegenseitig ausgeschlossen haben. Es handelt sich um OTRS::ITSM und OTRS::CiCS.
ITSM ist laut Entwickleraussage eine „ITIL konforme IT Service Management Lösung auf Open Source Basis“ [c]. Die Erweiterung ist modular aufgebaut und enthält neben dem Kern folgende Teilkomponenten [d]:

- IT Service Management
- Incident Management
- Problem Management
- Configuration Management & CMDB

Diese Komponenten ermöglichen im OTRS die Einbindung weiterer Funktionen, die sich an den Vorschlägen von ITIL orientieren. Insbesondere das Configuration Management ist als potentielle Quelle für umfassende Informationen zur technischen Infrastruktur der SLUB für diese Arbeit interessant.

Bei CiCS handelt es sich um eine Erweiterung, die von der Firma c.a.p.e. IT zur Verfügung gestellt wird [e]. Durch die Installation wird das Web-Interface von OTRS verändert und zusätzliche Funktionen integriert. An der SLUB wurde CiCS vor allem deshalb eingeführt, weil die Arbeit mit der neuen Oberfläche von den Mitarbeitern des Service Desks als ergonomischer empfunden wird. Da dies wichtiger als die Einführung der ITSM-Funktionen war, wurde die Einführung von OTRS::ITSM verschoben, bis beide Module miteinander kompatibel sind. Dies ist im März 2009 möglich geworden, als ebenfalls von c.a.p.e. IT das Erweiterungspaket OTRS::CiCS::ITSM veröffentlicht wurde. Diese Erweiterung wird nun auch an der SLUB eingeführt werden.

Das grundlegende Element von ITIL ist der Service. Auch im OTRS, insbesondere durch das ITSM-Paket, lassen sich Services definieren und nutzen. Auch an der SLUB Dresden werden Services aus einer eher technisch orientierten Sicht, unter der Bezeichnung *Verfahren*, angeboten und genutzt. Bisher wird der betroffene Service bei einem Incident, sofern er zu erkennen ist, nur im Ticket erwähnt, aber nicht explizit aus vordefinierten Angaben vorgegeben. Eine statistische Auswertung der Tickets, unter dem Kriterium der Häufigkeit des Auftretens bestimmter Verfahren, ist daher nicht möglich. Zudem ist an der SLUB auch kein explizites Servicedenken

vorhanden. Es gibt keine umfassenden Vereinbarungen zwischen Betreibern und Anwendern zum Betrieb der Verfahren. Für den Service Desk gibt es daher auch keine Vorgaben zu Bearbeitungs- und Lösungszeiten bestimmter Verfahren. Eine Definition, welcher Service von welchem Anwender genutzt wird, liegt ebenfalls nicht vor. Die Problematik setzt sich darin fort, dass auch die Anwender, die von ihnen genutzten Angebote der IT-Abteilung nur selten als Service auffassen. Bei der Meldung eines Incidents muss der Mitarbeiter an der Hotline daher versuchen herauszufinden, von welchem Service der Anwender redet. Wie gut das gelingt, hängt zum großen Teil von der Erfahrung und dem Wissen des Mitarbeiters ab. Wenn es nicht gelingt, bleibt die korrekte Zuordnung dem Second-Level-Support überlassen. Wenn dieser dabei feststellt, dass bisher gar kein oder das falsche Verfahren zugeordnet wurde, wird die Bearbeitungszeit des Incidents unter Umständen unnötig verlängert, da der Vorgang erst an einen anderen Verantwortlichen übergeben werden muss. In dieser Hinsicht ist allerdings keine generelle Neuausrichtung angedacht. Wenn Services im Support eine Rolle spielen sollen, dann vorwiegend zur Gliederung der vorhandenen technischen Infrastruktur.

2.3 Mögliche Verbesserungsschritte

Anhand der aufgezeigten Anforderungen und Probleme lassen sich verschiedene Wege ableiten, durch die die Arbeit des Service Desks verbessert werden kann. Eine einheitliche und strikte Ausrichtung nach ITIL ist allerdings nicht vorgesehen. Das liegt vor allem daran, dass dies ein extrem aufwändiges Vorhaben wäre, bei dem man schon auf Ebene des Managements ansetzen müsste. Die Anwender müssten entsprechend geschult werden und sich bewusst machen, welche Services sie während ihrer täglichen Arbeit nutzen. Anders lässt sich ein unternehmensweites Servicedenken nicht umsetzen. Stattdessen soll der Service Desk in die Lage versetzt werden, die Anliegen der Kunden besser einordnen zu können. Dies ist nur dann möglich, wenn weitergehende Informationen zu den Verfahren und ihren Komponenten zur Verfügung gestellt und die gewohnten Vorgänge nicht unnötig verkompliziert werden.

Ein möglicher Ansatz dafür ist die Zentralisierung der vorhandenen Informationsquellen in einem zentralen System. Anstatt in den verschiedenen Systemen suchen zu müssen, bietet es sich an, die relevanten Informationen ins

OTRS zu übernehmen. So ist es zum Beispiel möglich, mit dem ITSM-Modul eine eigene CMDB zu erstellen und FAQ-Einträge mit weiterführenden Informationen anzubieten. Dies würde das OTRS näher in Richtung eines von ITIL beschriebenen SKMS bringen. In dieser Hinsicht wäre auch die Nutzung des OTRS zur Pflege einer KEDB möglich. Wenn die Beschreibungen von Workarounds und Lösungen konsequent als FAQs den vorhandenen Tickets hinzugefügt werden, erleichtert dies bei wieder auftretenden Incidents die Bearbeitung. Teilweise geschieht dies schon in der Praxis. Allerdings existiert noch kein richtiges Konzept für Aufbau und Pflege der FAQs.
Ein weiterer potentieller Ansatz ist die Erstellung eines Configuration Models nach den Vorgaben von ITIL. Diese würde besonders unerfahrenen Mitarbeitern am Service Desk helfen, die Zusammenhänge in der technischen Infrastruktur des SLUB zu verstehen. Allerdings wäre dies ein Vorhaben von hohem zeitlichem und organisatorischem Aufwand.
Der Arbeitsprozess des Service Desks kann ebenfalls optimiert werden, indem man die Funktionen des BVZ vollständig im OTRS abbildet. Allerdings ist dies aufgrund unterschiedlicher Konzeptionen im Zweck und im Workflow der Systeme nur sehr schwierig möglich. Vor allem der Bezug von Configuration Items zu den Vorgängen, die in Tickets beschrieben werden, lässt sich nicht so umsetzen, wie es im BVZ der Fall ist.

Etwas, das alle diese Wege gemeinsam haben ist, dass eine technische Basis im OTRS geschaffen werden muss, damit sie umgesetzt werden können. Dies soll daher auch der erste Verbesserungsschritt und Ziel dieser Arbeit sein. Wichtigster Bestandteil ist dabei die Erstellung der CMDB, die als Informationsquelle für andere Komponenten des OTRS dient. Aber auch ein Konzept zur Strukturierung der gewonnenen Daten unter Verwendung vorhandener Verfahren spielt eine wesentliche Rolle. Wichtig ist dabei vor allem, dass möglichst viele nützliche Informationen in die CMDB einfließen, ohne dass übermäßig viel manueller Aufwand dafür notwendig wird. Wie von ITIL vorgeschlagen, soll daher ein automatisierter Abgleich mit mehreren Informationsquellen erfolgen. Die Weiterentwicklung dieses Konzepts zur Integration einer KEDB oder eines Configuration Models ist hingegen nicht Teil dieser Arbeit.

2.4 Lösungsansatz und Vorgehen

Um die in Abschnitt 2.3 beschriebenen Ziele zu erreichen, sind im Wesentlichen zwei Schritte notwendig. Als Erstes muss der Aufbau eines Konzepts zu Inhalt und Strukturierung der CMDB erfolgen. Danach wird ein System zum automatischen Abgleich und zur Strukturierung von CI-Daten geschaffen. Als Datenquelle zur Erstellung dienen dabei die in folgender Tabelle aufgeführten Systeme.

System	**Verwendete Daten**
BVZ	Ausgewählte Daten physischer Geräte
Virtual Center	Ausgewählte Daten virtueller Maschinen
Nagios	Überwachungsergebnisse

Tabelle 2: Verwendete Datenquellen

Als technische Basis für das System dient OTRS in der Version 2.3.4 unter Verwendung der Pakete ITSM 1.2.3, CiCS 2.1.1 und CiCS::ITSM. Die zu erstellenden Scripte werden so wenig wie möglich in die bestehende Systemkonfiguration eingreifen und als eigenständige Software arbeiten. Es ist nicht vorgesehen, dass die übernommenen Daten im OTRS gepflegt werden. Es erfolgt also eine reine Datenübernahme, die scriptgesteuert ergänzt wird. Alle Daten zu physischen Geräten werden weiterhin im BVZ gepflegt. Die Daten zu VMs und aus Nagios werden von jeweiligen Systemen selbst bereitgestellt und benötigen daher grundsätzlich keine manuelle Bearbeitung. Im Gegensatz zu den CI-Daten sind für die Erstellung eines Servicekonzeptes noch weiterführende organisatorische Betrachtungen notwendig. Das Konzept wird primär der Strukturierung und Informationsgewinnung dienen.

In der Arbeit wird zudem nur die technische und organisatorische Basis für die notwendigen Änderungen geschaffen. Alle Maßnahmen werden nur soweit durchgeführt, bis ihre Umsetzbarkeit in der Praxis nachgewiesen ist. Die letztendliche vollständige Umsetzung im Produktivsystem der SLUB ist für einen späteren Zeitpunkt vorgesehen.

3 Konzept und Erstellung der CMDB

3.1 Struktur

Prinzipiell ist im BVZ schon eine geeignete Grundstruktur vorhanden, auf der man eine CMDB für das OTRS aufbauen kann. Allerdings ist es nicht möglich, eine eins-zu-eins Abbildung zwischen den Systemen durchzuführen. Das liegt zum einen an dem technischen Aufbau der beiden Systeme und zum anderen an dem Bedarf von Anpassungen an der bestehenden Struktur. Welche technischen Maßnahmen getroffen werden müssen, wird aber erst im folgenden Abschnitt betrachtet.
Um die Daten aus dem BVZ ins OTRS zu bringen, ist es vor allem notwendig zu betrachten, welche Daten vorhanden sind und inwieweit diese genutzt werden sollen. In erster Linie bietet das BVZ eine vollständige Auflistung aller verwalteten Daten zu einem Gerät. Dazu zählen folgende Werte:

- Inventarnummer
- Seriennummer
- Bezeichnung
- Lieferdatum
- Garantieablauf
- Verwendungsstatus
- Verantwortlichkeit
- Standort
- Geräteklassen-spezifische Attribute
- Platznummern, IP-Adresse und Anwender bei Netzwerkgeräten

Diese Daten sind für die Verwendung am Service Desk von unterschiedlich starkem Interesse. So ist z.B. der Standort für die Durchführung von Wartungsmaßnahmen relevant, aber das Lieferdatum nur in Ausnahmefällen interessant. Einige der geräteklassenspezifischen Attribute werden nicht mit ins OTRS übernommen, da diese nicht mehr gepflegt werden oder keinen Nutzen für die Bearbeitung von Incidents haben. Zusätzlich werden auch diverse Daten, die nur haushaltsrechtlich relevant sind (z.B. Rechnungsnummern und Preise) nicht übernommen. Eine Übersicht über die Daten, die übernommen werden, findet sich in Anlage 1.

Wie man in der Anlage erkennen kann, sind Gerätedaten nach ihrer Klasse aufgeschlüsselt. Dies ist nicht nur durch die geräteklassenspezifischen Attribute aus dem BVZ begründet, sondern auch ein grundlegendes Konzept für den Aufbau der CMDB im OTRS. Auch in ITIL wird diese Form der Einteilung vorgeschlagen. Jedes Configuration Item muss einer Klasse zugeordnet werden, für die eine bestimmte Struktur aus Attributen vordefiniert ist. Details dazu werden im nächsten Abschnitt erläutert.
Es ist vorgesehen, dass die Klassen, die im BVZ existieren, erhalten bleiben. Sie müssen daher nach den Vorgaben der Attribute aus Anlage 1 im OTRS angelegt werden. Wie dies konkret erfolgt, wird ebenfalls erst im nächsten Abschnitt betrachtet. Zusätzlich zu den Klassen des BVZ, wird eine Klasse für virtuelle Maschinen eingeführt. Der Aufbau dieser Klasse wird im Zuge der Arbeit separat betrachtet.

Neben der Erstellung der CIs erfolgt eine Definition von Services, anhand der Verfahrensliste der IT-Abteilung der SLUB. Die Verfahrensliste umfasst in etwa 90 Einträge. Da dies relativ viele Services ergibt, sollte man vor der Umsetzung sinnvolle Zusammenfassungen und Kürzungen zu finden. Es ist allerdings wichtig, dass alle Verfahren übernommen werden, für die die Meldung eines Incidents denkbar ist. Eine Zusammenfassung ist nur dann sinnvoll, wenn ein besonders enger funktionaler Zusammenhang besteht. Die endgültige Definition, welche Verfahren letztendlich übernommen werden, würde im Rahmen dieser Arbeit zu weit führen und erfolgt daher zu einem späteren Zeitpunkt.
Das Konzept von OTRS sieht vor, die Auswahl eines Services an bestimmte Anwender zu binden. An der SLUB gibt es eine solche Eingrenzung allerdings nicht. Theoretisch könnte jeder der rund 250 Anwender einen Incident in Bezug auf einen der 90 Services melden. Auch eine Definition dieser Einschränkungen aus praktischen Gesichtspunkten ist nicht praktikabel, da dafür ein extrem hoher Arbeitsaufwand entstehen würde. Eine Zuordnung von Anwendern zu Services wäre nur dann sinnvoll, wenn man den Kreis der Anwender, die einen Service-Incident melden dürfen, auf wenige verantwortliche Mitarbeiter beschränken würde. Dieses Vorgehen erfordert allerdings weitgehende Vereinbarungen auf Ebene des Managements und ist daher nicht vorgesehen.

Allein für sich sind die Daten zwar interessant, allerdings ist es für die Arbeit am Service Desk von Vorteil sie zu verknüpfen. Dadurch lassen sich Zusammenhänge besser darstellen, Informationen schneller auffinden und mehr Ansätze für statistische Auswertungen finden. In welchen Formen diese Verknüpfung erfolgt, soll im Folgenden aufgezeigt werden.

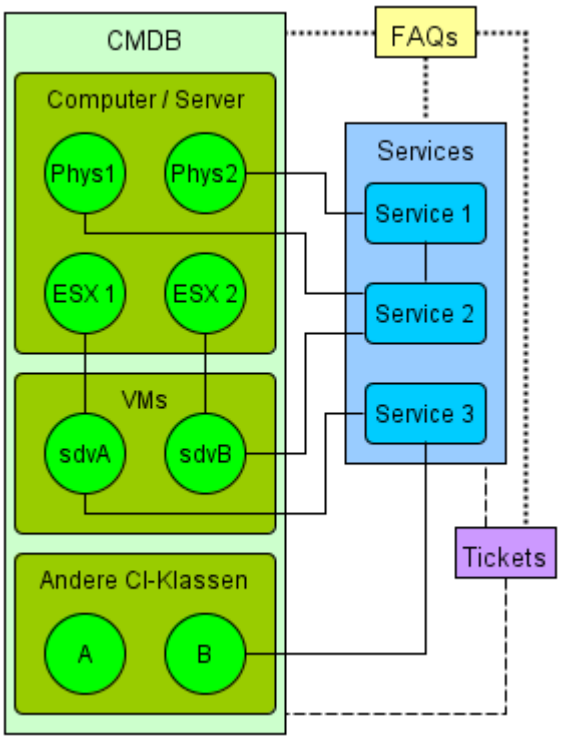

Abbildung 2: Struktur der CMDB und Bezug zu anderen Objekten

In Abbildung 2 sind die Formen der Verknüpfungen aufgeführt, die zur Strukturierung der neuen Daten beitragen. Unter den Geräteklassen aus dem BVZ ist keine weitere Strukturierung vorgesehen, da sich hier kaum sinnvolle Zusammenhänge finden lassen. Zwar gibt es viele Geräte, die miteinander verbunden sind, allerdings ist dies nicht im BVZ dokumentiert. Außerdem besteht nur selten ein wirklicher funktionaler Zusammenhang. Lediglich zwischen den virtuellen Maschinen und der physischen Maschine, auf denen diese jeweils laufen, ist eine Verknüpfung vorteilhaft. Dies dient dazu, Zusammenhänge zwischen physischen Defekten am Hostsystem und den resultierenden Auswirkungen auf der VM zu erkennen.

Eine weitere Form der Verknüpfung ist die Zuordnung von Configuration Items zu Services. Diese Informationen sind bereits weitestgehend im internen Wiki dokumentiert und müssen übertragen werden. Eventuell noch unvollständige Dokumentationen zu bestimmten Verfahren müssen zur Gewährleistung der

Vollständigkeit mit den notwendigen Informationen ergänzt werden. Da eine Automatisierung hier nur schwer möglich ist, erfolgt die Erfassung und Pflege auf manuellem Weg. Ein Service kann dabei mit einer beliebigen Anzahl von Cis verknüpft werden, bzw. auch ein CI mit einer beliebigen Anzahl von Services. Ziel dieser Verknüpfungen ist die Darstellung des Zusammenhangs zwischen dem abstrakten Service und der technischen Basis. So kann leichter erkannt werden, ob ein gerätebezogener Incident Auswirkungen auf einen Service hat.
Ein ähnliches Ziel verfolgt auch die Verknüpfung von Services untereinander. Diese Informationen sind ebenfalls weitestgehend im internen Wiki dokumentiert und müssen nur noch auf Vollständigkeit geprüft und übernommen werden. Sie helfen zu erkennen, ob ein Service-Incident eine tiefer liegende Ursache hat. So wird zum Beispiel eine Störung beim DNS-Service bei vielen anderen Services zu Problemen führen.

Die neu hinzukommenden Daten werden auch im Zusammenspiel mit bereits vorhandenen Informationen genutzt. Welche Möglichkeiten dabei bestehen, soll im Folgenden näher betrachtet werden.
Das Ticket ist das zentrale Element für die Bearbeitung jedes Incidents. Nach dem Servicegedanken von ITIL ist im OTRS auch die Auswahl eines Services für ein Ticket möglich. Wie schon erwähnt, können Services an Kunden gebunden werden. Wenn man diese Option, wie vom beschriebenen Konzept vorgesehen, nicht nutzt, werden die Services, die im Ticket zur Verfügung stehen sollen, stattdessen als Standard-Services für alle Anwender freigeschaltet. Dabei muss man bedenken, dass diese Services bei der Ticketerstellung aus einem einzigen Drop-down-Menü ausgewählt werden müssen. Wenn zu viele Services freigeschaltet sind, ergibt sich eine sehr lange Liste, in der der gewünschte Service gesucht werden muss. Die Liste ist zwar alphabetisch sortiert, aber wenn man den genauen Namen nicht weiß, hilft das bei der Suche nicht weiter. Eine sinnvolle Kategorisierung ist daher die zwischen häufig betroffenen und weniger häufig betroffenen Services. Wenn häufig betroffene Services am Anfang der Liste stehen, wird der Suchaufwand im Durchschnitt reduziert. Generell sollte es auch nicht verpflichtend sein, direkt bei der Annahme einer Benutzermeldung einen Service auszuwählen. Nur wenn dem Mitarbeiter schnell klar ist, welcher Service betroffen ist, wird dieser auch ausgewählt. Wenn dies nicht der Fall ist, kann immer noch eine Einordnung durch den Second-Level-

Support erfolgen. Die Auswahl des Services dient hauptsächlich dazu, das Ticket zu kategorisieren und die Möglichkeiten für Statistiken zu erweitern.
Eine Definition von Service Level Agreements, die auf den Services aufbauen, ist nicht vorgesehen, da es noch keine Vereinbarungen dieser Art für den Betrieb der Verfahren gibt. Durch diese könnten Reaktions-, Aktualisierungs- und Lösungszeiten vorgegeben werden. Perspektivisch ist es interessant zu betrachten, ob diese Vorgaben in bestimmten Teilbereichen für die Arbeit des Service Desks nützlich wären.

Unabhängig von allen Strukturen werden FAQ-Einträge verknüpft. Sie verweisen je nach Inhalt auf Tickets, Services oder Configuration Items, die für die jeweilige Fragestellung von Interesse sind. Dies erleichtert dem Betrachter des Eintrags den Zugang zu weiterführenden Informationen. Die Verknüpfung kann dabei auch als Basis für eine KEDB dienen. Ein Konzept, wie diese aufgebaut und gepflegt würde, existiert allerdings noch nicht.

3.2 Abgleich der Gerätedaten

3.2.1 Daten aus dem BVZ

3.2.1.1 Allgemeine Daten

Um Daten von einem System in ein anderes zu bringen, muss eine Schnittstelle geschaffen werden, über die der Datenaustausch kontrolliert und störungsfrei erfolgen kann. Der Abgleich der Daten zwischen OTRS und BVZ erfolgt nur in eine Richtung – vom BVZ ins OTRS. Sämtliche Änderungen der Gerätedaten erfolgen im BVZ. Eine Aktualisierung der CMDB im OTRS erfolgt erst, wenn ein neuer Datenabgleich erfolgt.
Für den Import der Daten ins OTRS ist bereits eine geeignete Schnittstelle vorhanden, die über das Import-Export-Paket von ITSM ins System eingebunden wird. Über diese kann sowohl der manuelle Import im Webinterface, als auch ein komplett automatisierter Import über ein entsprechendes Script erfolgen. Die Daten für den Import werden dabei aus einer CSV-Datei entnommen. Es ist wichtig die Anforderungen dieser Schnittstelle näher zu betrachten, um einen erfolgreichen Import zu erhalten.

Um überhaupt CIs in die CMDB einpflegen zu können, ist das Erstellen von Configuration Item-Klassen im *General Catalog* erforderlich. Vom System wird bei der Einrichtung des General Catalog-Pakets bereits eine hierarchische Klassenstruktur angelegt. Diese umfasst unter anderem *Computer*, *Hardware* und *Software* sowie entsprechende Untertypen, wie z.B. *Laptop*, *Desktop* und *Server* für die Klasse *Computer*. Für die Daten, die aus dem BVZ übernommen werden, ist diese Struktur allerdings nicht brauchbar. Von OTRS ist für diesen Fall vorgesehen, dass diese Vorgaben deaktiviert werden, indem man ihre Gültigkeit auf *ungültig* setzt und neue Klassen anlegt. Das Anlegen der neuen Klassen erfolgt gemäß den Vorgaben aus dem BVZ. Welche dies sind, ist in Anlage 1 aufgeführt. Untertypen sind für die Klassen nicht vorgesehen und müssen daher auch nicht angelegt werden.
Für jede neu angelegte Configuration Item-Klasse ist es notwendig, eine Definition der zugehörenden Attribute vorzunehmen. Dies geschieht über das Anlegen eines frei definierbaren Schemas, welches in Textform mit einer passenden Beschreibungssprache erstellt wird [f]. Im Wesentlichen handelt es sich dabei um eine beliebig lange Auflistung von Feldern, die unterschiedliche Eigenschaften annehmen können. Im einfachsten Fall wird ein einzeiliges Textfeld verwendet. Allerdings sind auch andere Feldtypen möglich. Dazu zählen:

- Drop-down-Menüs mit vordefinierten Werten
- Datumsfelder
- mehrzeilige Textfelder
- Verweis auf andere Werte aus den OTRS-Daten (z.B. auf Personen)

Sämtliche Felder können als Pflichtfelder oder als wiederholbar definiert werden. Ebenso ist es möglich Felder hierarchisch zu ordnen, also bestimmte Felder anderen unterzuordnen.
Zur Umsetzung der Attribute, die in Anlage 1 aufgeführt werden, sind meist einzeilige Textfelder ausreichend. Ausnahme sind zwei Datumsfelder für *Lieferdatum* und *Garantieablauf*. Für den Import der Daten spielt der Feldtyp keine besondere Rolle, außer dass eventuelle Einschränkungen durch vordefinierte Wertbereiche beachtet werden müssen. Interessanter sind die Feldtypen bei der Bearbeitung und Suche im Webinterface, da sie dort die Bedienung vereinfachen. Hierarchische Strukturen werden bei ausgewählten Attributen berücksichtigt, um die Übersichtlichkeit zu

verbessern. So wird zum Beispiel das Attribut *Taktrate* dem Attribut *Prozessortyp* untergeordnet. Pflichtfelder sind hingegen nicht notwendig, da die Daten nicht im OTRS editiert werden.
Neben den bereits erwähnten Feldeigenschaften können unter anderem noch folgende Einstellungen vorgenommen werden:

- Searchable – schaltet das Feld für die Suchfunktion frei
- Size – Länge des Feldes im Webinterface
- MaxLength – maximale Länge des Inputs
- CountMin/CountMax/CountDefault – Vorgaben für wiederholbare Elemente

Bei der Definition der neuen Klassenschemas werden alle Werte als *Searchable* freigegeben und auf sinnvolle Längen beschränkt. Wiederholbare Elemente sind nicht vorgesehen.
Während des Tests wurde nur das Schema für die Klasse *Computer* erstellt und befindet sich zur Anschauung in Anlage 2. Die Schemas für die anderen CI-Klassen müssen für den Einsatz im Produktivsystem noch nach den Vorgaben aus Anlage 1 und unter Beachtung der beschriebenen Aspekte erstellt werden. Diese Aufgabe hat keine weitere Bedeutung für die theoretische Betrachtung des Abgleichs. Daher erfolgt das Anlegen weiterer Schemas erst zu einem späteren Zeitpunkt.

Von besonderer Bedeutung sind die Felder für *Name*, *Verwendungsstatus* und *Vorfallsstatus*. Sie existieren für jedes CI und werden nicht im Schema der Klasse definiert. Da sie für den Import trotzdem eine wichtige Rolle spielen, werden sie im Folgenden genauer betrachtet.
Der Name ist die geläufige Bezeichnung für das CI. Für dieses Feld wird der Produktname, der im BVZ eingetragen ist, verwendet.
Weitere Anpassungen sind für den Verwendungsstatus notwendig. Dieser gibt an, in welchem Zustand der Nutzung sich ein CI befindet. Im BVZ ist diese Information vorhanden, ist aber in den verwendeten Begriffen nicht kompatibel mit den vorgegebenen Werten in OTRS. Deshalb müssen entweder die Status-Werte aus dem BVZ nach dem Export mit den Begriffen aus den OTRS-Vorgaben ersetzt oder kompatible Werte im OTRS definiert werden. Diese Möglichkeit ist von OTRS durchaus beabsichtigt und lässt sich direkt im General Catalog vornehmen. Negative Auswirkungen durch die Änderungen sind daher auch langfristig (z.B. beim Einsatz

neuer OTRS-Versionen) nicht zu erwarten. Da sich die momentan im BVZ verwendeten Begriffe im Praxiseinsatz bewährt haben, werden sie weiterhin verwendet. Das bedeutet, dass im General Catalog folgende Werte für den Verwendungsstatus (*DeploymentState*) neu angelegt werden müssen:

- inventarisiert
- im Einsatz
- in Reparatur
- ausgesondert
- befristet

Bereits vorgegebene Werte im OTRS werden durch die neuen Werte überschrieben bzw. auf ungültig gesetzt und stehen danach nicht mehr zur Verfügung.
Für den Vorfallsstatus gibt es kein Äquivalent im BVZ. Er wird für alle Geräte auf *Operational* gesetzt. Änderungen des Wertes erfolgen nur temporär durch die in Abschnitt 3.4 beschriebenen Maßnahmen im Zusammenhang mit der Verwendung von Ergebnissen aus der Netzwerküberwachung.

3.2.1.2 Identifikation der CIs und Mapping

Beim Import der Daten von Configuration Items ist es wichtig, einen Identifikator anzugeben. Dadurch wird es möglich die Attribute von bestehenden CIs bei Änderungen zu aktualisieren.

Ein mögliches Feld dafür ist im OTRS die *Number*. Wie Name, Verwendungs- und Vorfallsstatus ist auch dieses Feld standardmäßig für alle Klassen definiert. Innerhalb von OTRS ist die Number eine Art Identifikator eines jeden Configuration Items. Allerdings ist sie nicht der Identifikator, der in den Datenbankstrukturen OTRS zur Identifikation eines CIs genutzt wird. Dafür gibt es eine extra ID, die zum Beispiel bei Verlinken von Objekten referenziert wird.
Beim Erstellen eines CIs in der CMDB wird die Number vom System automatisch vergeben und ist bei der weiteren Arbeit mit den CIs zwar sichtbar, aber nur in Ausnahmefällen von Interesse. Beim Import der Daten spielt sie hingegen eine größere Rolle. Beim ersten Import eines CIs kann die Number prinzipiell erst einmal

vernachlässigt werden, da sie wie schon erwähnt vom System vergeben wird. Falls Gerätedaten wiederholt importiert werden, kann die vergebene Number beim Import als Identifikator mit angegeben werden. Sie dient dann dazu die CIs zu erkennen und verhindert, dass mehrere CIs für das gleiche Gerät entstehen. Wenn ein CI mit einer bereits vorhandenen Number importiert wird, werden alle seine Daten (sofern in der Import-Datei angegeben) aktualisiert, unabhängig davon, ob sie seit dem letzten Import geändert wurden oder nicht.

Die Frage, die sich beim wiederholten Import stellt, ist: Wie kann man den CIs in der CSV-Datei die im OTRS vergebene Number zuordnen? Bei nur wenigen Einträgen ist dafür ein manueller Eintrag denkbar. Für ein Gesamtvolumen an Geräten, wie es beim Abgleich mit dem BVZ zustande kommt, ist dieser Weg allerdings nicht praktikabel. Es muss daher ein separater Identifikator aus dem BVZ gewählt werden, über den unabhängig von der Number die Abbildung der Gerätedaten auf die vorhandenen CIs erfolgt.

Denkbar wäre dafür die Verwendung der ID aus der BVZ-Datenbank. Diese ist nicht nur eindeutig, sondern lässt sich aufgrund ihrer Eigenschaften auch vor dem Import direkt auf die Number im OTRS abbilden. Wann man die Liste der Geräte in aufsteigender Reihenfolge der BVZ-IDs sortiert, bleiben die Geräte stets an der gleichen Position in der Liste. Dies wird durch zwei Aspekte garantiert. Erstens erhalten neu eingetragene Geräte ihre BVZ-ID in aufsteigender Reihenfolge. Bei der Sortierung bilden die neuen Geräte das Ende der Liste und beeinflussen die Reihenfolge vorhandener Geräte nicht. Zweitens werden Geräte im BVZ nie gelöscht. Dadurch kommt es zu keiner Verschiebung der nachfolgenden Listenelemente. Für die sortierte Liste können nun schon vor dem Import die passenden Number-Werte, die das OTRS vergeben würde, per Script generiert werden. Dabei handelt es sich um eine einfache 10-stellige Zahl (z.B. 1335000001), die pro Element der Liste um eins inkrementiert wird. Durch die dritte und vierte Stelle der Zahl wird die ID der Klasse des CI gekennzeichnet. Die erste und zweite Stelle bleibt pro OTRS-Installation konstant. Dieses Vorgehen hat den Vorteil, dass kein zusätzliches und ansonsten nutzloses Feld für jedes CI angelegt werden muss. Dies wirkt sich positiv auf Speicherplatzbedarf und Übersichtlichkeit aus. Allerdings ist es dafür notwendig, die passenden IDs für die Klasse aus der Datenbank zu suchen und die Scripte zur Erstellung der Number entsprechend zu modifizieren. Insgesamt ist dieses Vorgehen daher zu umständlich und wird so nicht umgesetzt.

Ein einfacherer Weg ist die Verwendung der Inventarnummer. Diese ist eindeutig und ändert sich auch nie. Wenn es zu einem Garantiefall kommt, wird im BVZ lediglich die Seriennummer des Gerätes geändert und kein anderes Gerät mit anderer Inventarnummer eingesetzt. Auch beim entsprechenden Configuration Item im OTRS ändert sich daher nur die Seriennummer. Vorhandene Verlinkungen zu dem Configuration Item bleiben also korrekt. Sie nimmt ebenso keinen unnötigen Speicherplatz weg, da sie zusätzlich zur physischen Identifikation der CIs dient. Im Störungsfall findet der Anwender die Inventarnummer als Label auf dem Gerät.
Das Problem mit diesem Attribut ist, dass zurzeit 39 Geräte in BVZ ohne Inventarnummer eingetragen sind. Für diese Geräte muss noch eine passende Inventarnummer vergeben werden, damit sie in der CMDB aufgenommen werden können. Bis dahin werden Geräte ohne Inventarnummer vom Import-Script ausgefiltert. Da dieser Weg ansonsten keine Nachteile aufweist und die weitere Anpassung der Importscripte erspart, wird er für die Umsetzung genutzt.

Wie schon erwähnt wurde, aktualisiert das OTRS beim Import alle Daten der angegebenen CIs. Zusätzlich wird der alte Datensatz als *Version* hinterlegt. Das ist bei Attribut-Änderungen interessant, allerdings beinhaltet nicht jeder Import auch wirklich Änderungen. Wenn bei jedem Import alle CIs mit angegeben werden, führt dies nicht nur zu zahlreichen redundanten Informationen im Interface, sondern auch zu einer Belastung der OTRS-Datenbank. Abgesehen von *der Number* würde für alle Attribute ein neuer Datensatz angelegt. Für jede Version werden ca. 100 Byte für die Attributwerte benötigt. Allerdings ergibt sich durch die Art der Speicherung in der OTRS-Datenbank ein wesentlich größerer Datensatz. Im getesteten Fall fielen pro CI durchschnittlich rund 4 KB an Daten an[2]. Beim Import aller vorhandenen 4000 Geräte (Werte gerundet), ergibt sich pro Abgleich ein Datenvolumen von rund 16 MB. Je nach Frequenz des Abgleichs summiert sich das Volumen in der Datenbank auf nicht zu vernachlässigende Höhen.
OTRS bietet keine Konfigurationsmöglichkeiten, um das Speicherplatzproblem bzw. die redundante Anzeige von Versionen zu umgehen. Die komplette Löschung aller CI-Daten vor jedem Import ist zwar technisch möglich, allerdings kommt es nach dem Import zu unvermeidbaren Problemen. Wenn eine Klasse nach dem neuen Import mehr CIs hat, als vor der Löschung, verschieben sich die Zuordnungen zwischen

[2] Testfall: Import von 1011 CIs der Klasse *Computer*. Größenänderung der Tabelle *xml_storage*: 4 MB.

Datenbank-ID und dem zugehörigen CI in der CMDB. Das führt bei Verlinkungen dazu, dass mit einem Objekt plötzlich andere CIs verknüpft sind als zuvor. Dadurch wird letztendlich ein wichtiger Teil des Konzepts zu Strukturierung der CMDB zerstört und es tauchen im ungünstigsten Fall falsche Informationen im System auf.
Die Lösung für dieses Problem besteht darin, bereits beim Export eine Filterung auf CIs mit geänderten Gerätedaten vorzunehmen. Während beim ersten Export alle Geräte ausgewählt werden, erfolgt bei den folgenden Exporten eine Beschränkung auf jene Geräte, die einen Änderungseintrag in einem bestimmten Zeitintervall aufweisen. Zwar wird für die geänderten CIs auch eine komplett neue Version für alle Attribute angelegt, aber für den Großteil der unveränderten CIs kommen keine zusätzlichen Daten ins System. Der Speicherplatzbedarf wird daher stark reduziert.
Eine weitere geringfügige Verbesserung im Bezug auf den verwendeten Speicher ergibt sich durch den Einsatz des MySQL Wartungsbefehls *Optimize Table* [g]. Dieser Befehl verbessert das Speicherverhalten bei langen Zeichenketten vom Typ *varchar*, wie sie durch die Versionsdaten der CIs im OTRS entstehen, indem ungenutzte Speicherbereiche freigegeben und die Daten defragmentiert werden. Dadurch wurde bei ersten Tests eine Verringerung des importierten Datenvolumens von ca. 20% erzielt. Die Einbindung des Befehls in die Import-Scripte ist daher vorgesehen.

Unabhängig vom Weg des Imports muss im OTRS ein *Mapping-Template* erstellt werden, dass die Daten einer CSV-Datei in die Attribute der CIs wandelt. Jedes Template kann nur für eine CI-Klasse definiert werden. Für diese Klasse wird angegeben, welches Feld eine Spalte in der CSV-Datei repräsentiert und in welcher Reihenfolge diese vorliegen. Zudem kann festgelegt werden, welches Feld der Identifikator ist. Aufgrund des oben beschriebenen Konzeptes ist diese Markierung beim Feld Inventarnummer vorgesehen. Alle Templates werden so aufgebaut, wie die entsprechende Klasse definiert wurde. Jedes Attribut ist genau einmal in der vorgegebenen Reihenfolge in der CSV-Datei vorhanden und wird dementsprechend so auf die Felder der CIs gemappt. Ausnahme ist das erste Feld im Template. Hier würde sich nach dem beschriebenen Konzept der Eintrag für die Number befinden, welche aber nicht verwendet wird. Allerdings darf auf das erste Feld auch kein anderer Wert gemappt werden, da hier in der CSV-Datei noch der nicht benötigte Identifikator aus dem BVZ liegt. Das Feld wird daher einfach unbesetzt gelassen.

Über Filter können CIs gezielt blockiert werden, wenn sie angegebene Kriterien erfüllen. Für den Import aus dem BVZ ist dies allerdings nicht vorgesehen, da es keine Notwendigkeit für weitere Eingrenzungen gibt. Zusätzlich werden noch allgemeine Informationen zu Zeichensatz und Trennzeichen des CSV festgelegt. Für die hier verwendeten Daten ist UTF-8 und das Semikolon zu wählen.
Wenn das Template vorliegt, können die Daten aus dem BVZ für die entsprechende Geräteklasse importiert werden. Dazu ist es notwendig eine CSV-Datei zu erstellen, die alle passenden Daten in korrekter Reihenfolge und Formatierung beinhaltet. Da ein manuelles Erstellen der Datei praktisch nicht umsetzbar ist, muss dieser Vorgang automatisiert werden.
Am sinnvollsten ist es, nicht nur die CSV-Datei zu erzeugen, sondern auch deren Import im gleichen Zug automatisch zu festgelegten Zeiten durchzuführen. Wie das erreicht werden kann, wird im Folgenden betrachtet.

3.2.1.2 Automatischer Abgleich

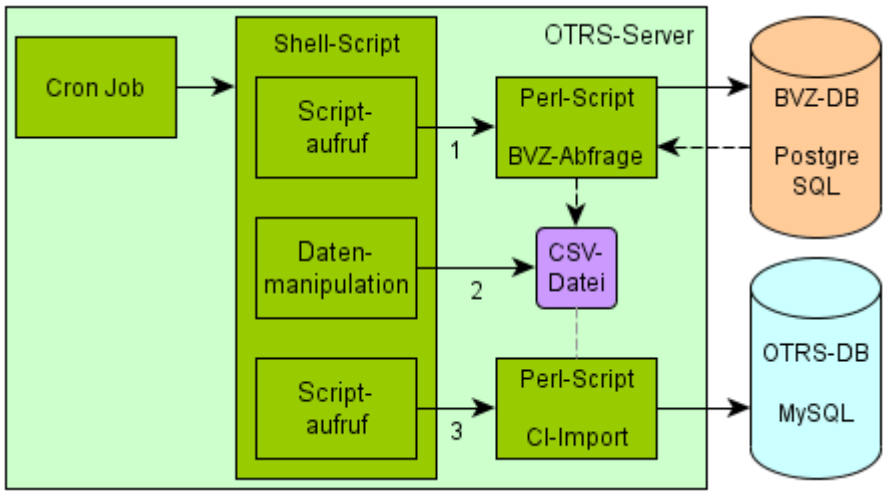

Abbildung 3: Datenabgleich bei physischen Geräten

In Abbildung 3 ist ein Modell für die automatisch gesteuerte Übertragung der Daten vom BVZ ins OTRS dargestellt. Der gesamte Prozess wird in bestimmten Abständen durch einen *Cron Job* angestoßen. Dieser ruft ein *Shell Script* auf, das die Ausführung aller folgenden Prozesse startet.
Diese Prozesse sind in Abbildung 3 mit den Ziffern 1, 2 und 3 markiert und werden in folgender Aufzählung näher beschrieben:

1. Aufruf eines Perl-Scripts zur BVZ-Datenbankabfrage
    - Verbindungsaufbau zu PostgreSQL-Datenbank [h]
    - Ausführung definierter SQL-Abfragen
    - Erhaltene Gerätedaten als CSV-Datei ablegen

2. Manipulation der Daten in der erhaltenen CSV-Datei
    - Einfügen des Standard-Vorfallsstatus
    - Zusammenfassen von Vor- und Nachname
    - Ersetzen der Bezeichnung für fehlende Geräteattribute
    - Löschen von Einträgen ohne Inventarnummer

3. Aufruf des Perl-Scripts zum Import in die OTRS-Datenbank
    - Nutzung der Import-Export-Schnittstelle von OTRS::ITSM
    - Angabe des passenden Import-Templates und der CSV-Datei
    - Auslesen der Daten der CSV-Datei und Schreiben in die OTRS-Datenbank

Der Import ist in den hier vorgenommenen Betrachtungen auf eine CI-Klasse beschränkt. Dies liegt daran, dass je nach Klasse andere SQL-Abfragen und Import-Templates verwendet werden müssen. Für den Import aller Klassen müssen im Script weitere Aufrufe für alle Schritte sowie neue Perl-Scripte zur BVZ-Abfrage angelegt werden.

In Anlage 3 befindet sich das Shell-Script, über das der Gesamtprozess ausgeführt wird. Er beinhaltet die Scriptaufrufe und die Datenmanipulation. Das Script zur Abfrage der Daten aus der PostgreSQL-Datenbank befindet sich in Anlage 4. Die konkrete SQL-Abfrage (für die Klasse Computer) wurde zur besseren Übersichtlichkeit in Anlage 5 ausgelagert. Aufgrund der komplexen Struktur der BVZ-Datenbank ist die Abfrage sehr umfangreich. Sie gliedert sich in drei wesentliche Teile, die letztendlich im Zusammenspiel für jedes Gerät im BVZ einen Datensatz mit allen relevanten Attributen erzeugen.
Der erste Teil greift auf die zentrale Gerätetabelle der BVZ-Datenbank zu und entnimmt dort aufgeführte allgemeine Informationen wie Name, IP-Adresse und Kommentare. Diese werden über *Joins* mit passenden Daten zu Standort und verantwortlichem Mitarbeiter ergänzt.

Im zweiten Teil werden für jedes Gerät die zugehörenden Attribute und deren Wert erfasst. Dabei ergibt sich allerdings aufgrund des Aufbaus der BVZ-Datenbank ein strukturelles Problem. Die für das Gerät aufgeführten Daten liegen nicht mehr in einer Zeile vor, sondern je nach Anzahl der Attribute in mehreren. Dies sieht dann zum Beispiel wie folgt aus:

Geräte-Nummer	Attribut	Wert
1	Größe	19 Zoll
1	Technologie	Röhre
1	Anschluss	Analog

Tabelle 3: Ausschnitt aus der Tabelle für Geräteattribute in der BVZ-DB

Notwendig ist allerdings folgende Form:

Geräte-Nummer	Größe	Technologie	Anschluss
1	19 Zoll	Röhre	Analog

Tabelle 4: Für den Import benötigte Form der Geräteattribute

Diese Form wird dadurch erreicht, dass man die Tabelle transponiert. Konkret muss für jede Gerätenummer die Spalte *Wert* in eine Zeile umgewandelt werden. Die Zeile *Attribut* dient dann als Bezeichnung für die Spalten, in der gleichartige Attribute aufgeführt sind. Erreicht wird dies durch die Verwendung des *Case*-Ausdrucks [i] [j]. Dabei wird in der Projektion für jedes mögliche Attribut ein Case-Ausdruck erstellt, unter dem alle existierenden Werte zu diesem Attribut in einer Spalte zusammengefasst werden. Wenn für ein Gerät ein Wert zu dem entsprechenden Attribut gefunden wurde, wird er in die Spalte eingetragen, ansonsten wird ein Platzhalter gesetzt.

Im dritten Teil wird der aktuelle Verwendungsstatus des Gerätes ermittelt. Im BVZ sind alle Statusänderungen verzeichnet. Um den aktuellen Status zu ermitteln, ist es notwendig für jedes Gerät genau die Statusänderung zu wählen, die das neuste Datum aufweist. Da die Datumsangaben als numerischer Wert verstanden werden

können, kann dies durch die Verwendung der Max-Funktion erreicht werden. Zusätzlich erfolgt in diesem Teil die Filterung auf die in einem bestimmten Zeitabschnitt geänderten Geräte. Dabei wird der Zeitabschnitt als Differenz zwischen dem aktuellen Zeitpunkt der Ausführung und einem Intervall von einer bestimmten Anzahl von Tagen berechnet. Dabei wird auf verschiedene Zeit- und Datumsfunktionen von PostgreSQL zurückgegriffen [k]. Für den Zeitabschnitt der letzten sieben Tage sieht der Ausdruck wie folgt aus:

```
datum > now() – interval '7 days'
```

Eine Optimierung dieses Aufrufs wird erreicht, wenn man statt des fest vorgegebenen Intervalls den Zeitpunkt des letzten Imports ermittelt. So bleiben die Daten auch aktuell, wenn der regelmäßige Datenabgleich durch eine Störung unterbrochen wurde. Eine entsprechende Anpassung wird noch vor der praktischen Umsetzung im Produktivsystem vorgenommen.
Aus administrativer Sicht ist bei der Durchführung des Imports nicht viel zu beachten. Generell muss nach dem ersten Import stichprobenartig untersucht werden, ob alle Klassen wie erwartet mit CIs gefüllt wurden und ob die Attribute vollständig sowie in richtiger Reihenfolge vorhanden sind. Nach dem zweiten Import ist sicherzustellen, dass keine CIs doppelt angelegt wurden. Wenn während dieser Schritte keine Probleme entstanden sind, brauchen weitere Routine-Checks nur noch bei Bedarf durchgeführt werden. Da es sich um einen Prozess mit relativ vielen Datenbankabfragen handelt, kann es bei Netzwerkproblemen zu Fehlern im Ablauf kommen. Allerdings wird der Datenbestand in der CMDB nicht beeinträchtigt, wenn der Import scheitert.

Für den Import ist es zusätzlich wichtig zu betrachten, wann und wie oft dieser durchgeführt wird. Da es dabei zu einer erhöhten Belastung von OTRS und dem Datenbanksystem kommt, ist ein Import während der normalen Arbeitszeiten ungünstig. Daher ist es besser, den Import in der Nacht durchzuführen. Ein besonders guter Zeitpunkt ist nach dem täglichen Backup der OTRS-Datenbank. Wenn es bei der Durchführung des Imports zu einem schwerwiegenden Fehler kommt, kann so auf die aktuelle Datenbankversion des Vortages zurückgegriffen werden. Der Datenverlust ist dadurch so gering wie möglich.

In der Testumgebung dauert ein Import von 1000 CIs etwa drei Minuten. Wenn nur wenige geänderte CIs ins Produktivsystem importiert werden, kann man davon ausgehen, dass der ganze Vorgang in wenigen Sekunden erledigt ist. Beim ersten vollständigen Import aller 4000 Geräte ist hingegen mit einer längeren Dauer zu rechnen.

Zur Frequenz des Abgleichs muss man beachten, wie oft Änderungen im BVZ erfolgen und wie wichtig die Änderungen für die Arbeit des Service Desks sind. Wenn man nach der Beschaffung bzw. dem damit verbundenem Anlegen neuer Geräte geht, reicht ein wöchentlicher Abgleich völlig aus. Wenn man die Änderungen bei vorhanden Geräten zeitnah erfassen will, ist allerdings ein täglicher Abgleich besser. Bei einem Blick in die Datenbank des BVZ kann man erkennen, dass bis auf wenige Ausnahmen täglich an mindestens einem Gerät Datenänderungen erfolgen. Für den Servicedesk sind diese geänderten Daten aber nicht zwangsläufig sofort interessant. Wenn es zu einer Änderung kommt, wird diese vom Service durchgeführt und danach das Gerät entsprechend auf seine Funktionsfähigkeit geprüft. Incidents in dem Zusammenhang sind daher eher selten zu erwarten. Rein technisch spricht allerdings nichts gegen einen täglichen Abgleich. Daher wird er auf diesem Weg umgesetzt.

Ein interessanter Aspekt ist, dass der Import nicht unbedingt immer automatisch durchgeführt werden muss. Wenn größere Änderungen im BVZ vorgenommen wurden, kann das Import-Script zusätzlich zu jeder beliebigen Zeit manuell angestoßen werden, um einen Abgleich außerhalb des normalen Intervalls zu ermöglichen.

3.2.2 Datenimport und Verknüpfung von virtuellen Maschinen

3.2.2.1 Technischer Ablauf

Virtuelle Maschinen werden im Gegensatz zu physischen Geräten nicht im BVZ verwaltet. Dies liegt zum einen an den unterschiedlichen Eigenschaften und zum anderen am Umgang mit den VMs. Unterschiede bestehen im Fehlen von Verwaltungsattributen wie Preis und Lieferdatum sowie Standort. VMs werden auch nicht ausgesondert, sondern gelöscht, wenn sie nicht gebraucht werden. Änderungen in der Konfiguration erfolgen meistens undokumentiert.

Der aktuelle Zustand jeder VM ist allerdings durch ausführliche Informationen im Virtual Center beschrieben. Um diese Daten nun in der CMDB nutzen zu können, ist es notwendig, eine passende Schnittstelle für sie zu finden. Dafür gibt es mehrere Möglichkeiten, die im Folgenden näher betrachtet werden.

Ein einfacher Ansatz wäre die Nutzung der Informationen, die auf jedem ESX-Server durch den Befehl *vmware-cmd –l* abgefragt werden können [l]. Über diesen Befehl erhält man eine Liste, in der alle dem Server zugeordneten VMs aufgeführt sind. Diese Liste kann per Cron Job erstellt und über E-Mail an den OTRS-Server gesendet werden, wo sie weiterverarbeitet wird. Ein Eintrag sieht zum Beispiel wie folgt aus:

```
/vmfs/volumes/48f6fd77-9fa9b6d5-9912-
0017a48f31aa/niebling_otrs/niebling_otrs.vmx
```

Der Eintrag stellt dabei den Pfad dar, unter dem die VM liegt. Der erste Teil des Pfads ist das verwendete Festplatten-Volume, welches als Information für Servicedesk kaum relevant ist und deswegen ignoriert werden kann. Interessant ist der eigentliche Dateiname. Dieser beinhaltet aufgrund der gewählten Namenskonventionen zwei relevante Informationen. Zum einen den Namen des Verantwortlichen vor dem Unterstrich und zum anderen den Hostname danach. Über Shell-Scripte können diese Informationen in eine CSV-Datei umgeschrieben werden, die für den Import ins OTRS geeignet ist.
Da der Hostname meist auch der Servername im DNS ist, lässt sich die IP über entsprechende Scriptbefehle problemlos ergänzen. Allerdings ist nicht jeder Eintrag so gut strukturiert, wie der im Beispiel. So haben zahlreiche Projektserver nur zusammengefasste Namen, wie z.B. *sdvdbodfarm*, die gesondert behandelt werden müssen. Zusätzlich existieren einige Server, die zwar über eine IP-Adresse verfügen, aber keinen DNS-Eintrag besitzen.
Aufgrund des Aufwands, der nötig ist, um die Listen in geeignete CSV zu wandeln und den relativ wenigen Informationen, die man dafür erhält, lohnt sie die Umsetzung dieses Weges nicht.

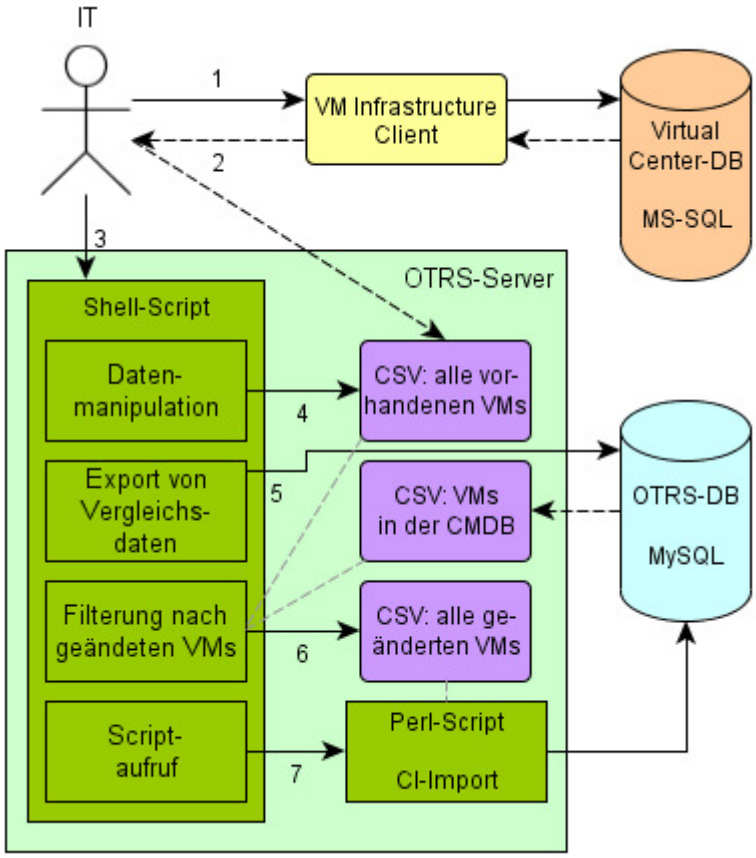

Abbildung 4: Datenabgleich bei virtuellen Maschinen

Ein weiterer Weg ist das in Abbildung 4 dargestellte manuelle Exportieren der Daten über den *VMware Infrastructure Client*. Dort ist es möglich, sich eine komplette Auflistung aller VMs geben zu lassen (Schritt 1 in Abbildung 4). Dabei können zahlreiche zusätzliche Attribute zu- und weggeschaltet werden. Über diese Attribute kann auch eine auf- und absteigende Sortierung erfolgen. Neben dem Namen gibt es Angaben zu IP, DNS-Eintrag, Betriebssystem und Eigenschaften der virtuellen Hardwarekomponenten. Wenn man die gewünschte Darstellung der VM-Daten eingestellt hat, können diese per Export in eine CSV-Datei geschrieben werden (Schritt 2 in Abbildung 4). Wichtig ist dabei, dass der Zeichensatz nach dem Export auf ISO-8859-1 geändert wird, da es sonst beim Import ins OTRS zu Formatierungsproblemen kommt.

Dieses Vorgehen hat den Vorteil, dass es sehr einfach ist und viele Daten erfasst werden können. Als Nachteil muss man sehen, dass die Erzeugung der CSV-Datei und deren Übertragung an den OTRS-Server manuell ausgelöst werden müssen.

An die gleichen Daten gelangt man auch, wenn man die Datenbank des Virtual Centers direkt abfragt. Dies kann vollautomatisch geschehen und erspart damit das manuelle Eingreifen. Zudem beseitigt dieser Weg einige Probleme, die sich beim

manuellen Export durch den vorgegebenen Zeichensatz ergeben. Alle relevanten Daten liegen zentral in der Tabelle *VPX_VM*, wodurch umständliche SQL-Abfragen vermieden werden [m]. Allerdings ist, im Gegensatz zur Verbindung auf den PostgreSQL-Server des BVZ oder den MySQL-Server des OTRS, der Zugriff auf die Virtual Center-Datenbank nicht ganz so einfach. Die Daten liegen auf einem MSSQL-Server. Um unter Linux mit einem Perl-Script darauf zugreifen zu können, sind zahlreiche Konfigurationsschritte notwendig [n]. Da keine direkte Verbindung aufgebaut werden kann, muss man den Umweg über ODBC gehen. Dafür müssen erst Treiber installiert und entsprechend konfiguriert werden. Dieser Prozess ist im Vergleich zu den dabei gewonnenen Ergebnissen unverhältnismäßig zeitaufwendig, sodass er im Rahmen der Diplomarbeit nicht weiter betrachtet wird. Eine Umsetzung dieses Schrittes ist aber trotzdem für einen späteren Zeitpunkt vorgesehen. Bis dahin werden die VM-Daten über den manuellen Export in die CMDB eingepflegt.

3.2.2.2 Attribute und Identifikator

Interessant ist nun die Frage, welches Attribut einer VM als Identifikator dient. Im Gegensatz zum BVZ gibt es keinen Identifikator, der beim Anlegen einer neuen Maschine inkrementiert wird. Es gibt nur einen *Universally Unique Identifikator* (UUID), der aber keine zeitlichen Abhängigkeiten aufweist. Eine Abbildung von der ID auf die Number ist hier also technisch gar nicht möglich. Daher muss ein Attribut gewählt werden, das wie die Inventarnummer einmalig und dauerhaft zugeordnet ist. Die Verwendung des UUID wäre möglich, ist aber als Information in der CMDB nicht interessant. Denkbar ist auch die Verwendung des Namens der VM. Dieser ist zwar einmalig, kann jedoch geändert werden und ist daher kein zuverlässiges Identifikationsmerkmal. Allerdings kommen Namensänderungen relativ selten vor. Wenn es doch zu einer Namensänderung kommt, ist dies auch kein großes Problem für die Strukturen der CMDB. Es wird lediglich ein neues CI angelegt, was keine größere Speicherbelastung nach sich zieht. Noch vorhandene Links verweisen dann auf das alte CI. Wenn die Umbenennung aufgrund einer grundlegenden Funktionsänderung des Servers stattfindet, ist dies sogar von Vorteil. Bei einer Umbenennung zur Korrektur eines Schreibfehlers ist diese Eigenschaft aber eher von Nachteil und wichtige Links müssten gegebenenfalls manuell geändert werden. Sofern sich dieses Vorgehen in der Praxis als nicht zu problematisch erweist, wird daher als Identifikator der Name der VM verwendet.

Wert	Export	Grund
Name	Ja	Bezeichnung der VM, Verwendung als ID
State	Ja	Verwendungsstatus (aus- / angeschaltet)
DNS Name	Ja	Serveradressierung
IP Address	Ja	Serveradressierung
Host	Ja	Zuordnung zu physischer Maschine
Notes	Ja	Informationen über Zweck der VM
UUID	Nein	irrelevant, nur theoretische Alternative für ID
Status	Nein	Unzuverlässig, kein zeitnahes Update möglich
Host CPU - MHz	Nein	kein zeitnahes Update möglich
Host Mem - MB	Nein	kein zeitnahes Update möglich
Guest Mem - %	Nein	kein zeitnahes Update möglich
Guest OS	Nein	Unzuverlässig, selten relevant
Memory Size – MB	Nein	selten relevant
CPU Count	Nein	irrelevant, überwiegend ein Kern
NIC Count	Nein	irrelevant, immer nur eine Netzwerkkarte
Uptime	Nein	selten relevant
VMware Tools State	Nein	selten relevant

Tabelle 5: Auswahl der Attribute für virtuelle Maschinen

Für den manuellen Export müssen nun die laut Tabelle 5 geforderten Daten in der Übersicht im VMware Infrastructure Client ausgewählt werden. Warum diese Daten gewählt wurden, ist als Begründung mit angegeben. Die Reihenfolge der Attribute kann bei der Anzeige eingestellt werden. Sie ist wie in Tabelle 5 vorgegeben zu wählen und wird auch in dieser Form vom Import-Template erwartet. Eine Sortierung ist nicht notwendig.

Nachdem die Liste exportiert und auf den OTRS-Server kopiert wurde, müssen die Daten der CSV-Datei noch verarbeitet werden. Dies kann wie beim Import der BVZ-Daten über ein Script geschehen, das durch einen Cron-Job gestartet wird. Solange der Import noch manuell erfolgt, ist nicht garantiert, dass bei Ausführung des Cron-Jobs bereits eine aktuelle CSV-Datei mit neuen Daten vorliegt. Das Script ist daher vorerst nach dem Einspielen der CSV-Datei manuell zu starten (Schritt 3 in Abbildung 4).

Durch das Script werden als Erstes folgende Änderungen an der eingespielten CSV-Datei durchgeführt (Schritt 4 in Abbildung 4):

- Löschen unnötiger Zeilen
- Ändern der Trennzeichen (OTRS kann keine Kommas verarbeiten)
- Ergänzen des Standard-Vorfallsstatus
- Ändern des *State* auf Verwendungsstatus (*Powered On* wird zu *im Einsatz*, *Powered Off* wird zu *inventarisiert*)
- Kürzung des *Host* (z.B.: *sdvvmesx01.slub-dresden.de* wird zu *sdvvmesx01*)

Ein zentrales Problem beim Import ist zu erkennen, welche VMs geänderte Daten aufweisen, um nur gezielt diese zu importieren. Anderenfalls wird auch hier das Problem der mehrfach redundanten Versionsdaten auftreten. Im Gegensatz zum BVZ existiert im Datensatz dafür keine Information. Es ist also ein skriptgesteuerter Vergleich zwischen vorhandenen Daten und der zu importieren CSV-Datei notwendig. Dafür wird vom Script zuerst ein Export der CIs für die Klasse der VMs gestartet (Schritt 5 in Abbildung 4). Hier wird ebenso die Import-Export-Schnittstelle des OTRS sowie das für den Import der VMs verwendete Template verwendet. Die erhaltenen Daten werden aufbereitet und in einer weiteren CSV-Datei abgelegt. Anhand der Daten der beiden CSV-Dateien wird verglichen, für welche CIs sich Daten geändert haben, welche VMs neu hinzugekommen sind und welche gelöscht worden. Der Datensatz dieser CIs wird ausgefiltert und in der für den Import vorgesehenen CSV-Datei abgelegt (Schritt 6 in Abbildung 4).
Nach diesen Schritten werden die Daten über die Import-Export-Schnittstelle importiert (Schritt 7 in Abbildung 4). Wie beim Import der BVZ-Daten wird dabei auf ein vordefiniertes Schema zugegriffen. Der Aufbau ähnelt dem der BVZ-Schemas, beschränkt sich aber auf einfache Textfelder, die mit den in Tabelle 3 aufgeführten Attributen belegt werden. Das Shell-Script, mit dem die beschriebenen Prozesse ausgeführt werden, befindet sich in Anlage 6.

3.2.2.3 Verlinkung

Nach dem Import ist entsprechend des Konzepts der CMDB eine Verlinkung zwischen der VM und dem physischen Server, auf dem sie läuft, vorgesehen. Da es keine Importfunktion für Links gibt, bietet sich eine direkte Manipulation der Datenbank an. Dazu muss zuerst analysiert werden, wie Links in der Datenbank abgelegt werden. Dies geschieht in der Tabelle *link_relation*, die wie folgt aufgebaut ist.

source_object_id	source_key	target_object_id	target_key	type_id	state_id	create_time	create_by
3	1	3	2	6	1	2009-06-30 09:07:46	23

Tabelle 6: Definition eines Links in der Datenbank des OTRS

Die einzelnen IDs, die in Tabelle 4 aufgeführt sind, können je nach System und den vorhandenen Definitionen abweichen. Ein neuer Link kann einfach durch Anlegen eines neuen Tabelleneintrags erfolgen. Die Object-IDs bleiben immer gleich und Verweisen auf die Objektklasse Configuration Items. Interessanter sind Source- und Target-Key, über die die IDs der zu verlinkenden CIs referenziert werden. Die IDs von Source und Target, also der VM und dem physischen ESX-Server kann man in der Tabelle *configitem* finden. Die Type-ID beschreibt den Verlinkungstyp und die State-ID die Gültigkeit des Links. Ebenfalls angegeben sind Informationen zu Erstellungszeit und dem verantwortlichen Ersteller. Ein weiterer Aspekt, der beachtet werden muss, ist die Auswahl des Link-Typs. Im OTRS stehen mehrere Typen zur Auswahl, die teilweise auch funktionale Auswirkungen bei Vorfallsstatusänderungen haben. Allerdings gibt es keinen Typ, durch den sinnvolle die Auswirkungsbeziehungen zwischen VM und ESX-Server dargestellt werden können. Für diese Beziehung eignet sich daher am ehesten der Typ *ConnectedTo*. Bei diesem hat eine Statusänderung bei einem Gerät keine Auswirkungen auf andere Geräte.

Um die gewünschten Links automatisch erzeugen zu können, ist es notwendig, für jede VM einen SQL-Befehl per Script auszuführen. Für jeden Befehl sind dabei die entsprechenden IDs in der OTRS-Datenbank für die CIs zu ermitteln. Ein geeigneter Weg ist dabei die Manipulation der CSV-Datei, über die der Import der VMs erfolgt.

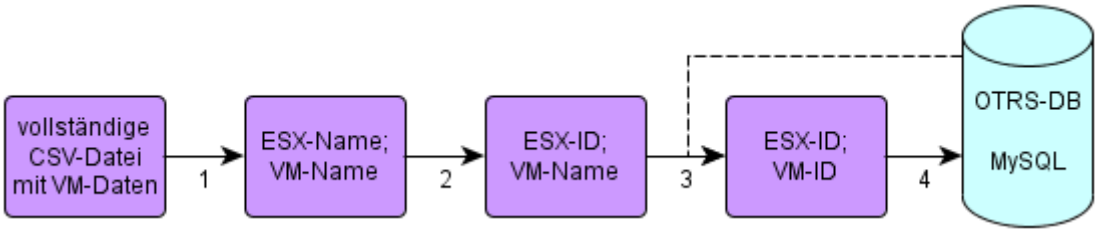

Abbildung 5: CSV-Manipulation zur Vorbereitung der Erzeugung von Links

Diese wird, wie in Abbildung 5 dargestellt, über Scriptbefehle wie folgt manipuliert:

1. Filtern der CSV-Datei auf Werte für den ESX-Host-Name und VM-Name
2. Ersetzen von ESX-Name durch passende ID des CI in OTRS
3. Abfrage der ID des CI mit passenden VM-Namen in der OTRS-DB, Ersetzen des Namens durch die ID
4. Einsetzen der ermittelten Daten in SQL-Befehl, Schreiben des Datensatzes zur Erzeugung der Links in die Datenbank

Im Gegensatz zum Anlegen der VM-IDs in Schritt drei, lassen sich die IDs der ESX-Server in Schritt zwei nicht so einfach aus der OTRS-Datenbank auslesen. Dies liegt zum Teil am Aufbau der Datenbank bzw. am gewählten Mapping beim Import. Während bei den VMs Name und ID über die Tabelle *configitem_version* sehr einfach zugeordnet werden können, ist dies für die ESX-Server nicht möglich. Für diese ist in der Tabelle als Name nur das Modell des Gerätes hinterlegt. Der eigentlich benötigte Wert (Platznummer) ist hingegen nur über stark verschachtelte Abfragen zu finden. Abgesehen vom benötigten Aufwand, würde eine Abfrage auch nicht zum Erfolg führen, da zurzeit nur für 5 der 14 ESX-Server der Name im BVZ hinterlegt ist. Es ist daher angebracht und auch praktikabel die IDs manuell zu suchen. Dies kann im OTRS-Interface erfolgen, indem man die angelegte CMDB nach den Inventarnummern der ESX-Server durchsucht. Die ID findet man dann in der Link-URL zum Configuration Item. Für die vorhandenen Server sind im angelegten Script bereits die passenden IDs eingetragen. Nur wenn es zu

Änderungen an der Serverstruktur (z.B. bei der Anschaffung neuer ESX-Systeme) kommt, müssten diese aktualisiert werden.
Das verwendete Script löscht zudem bei jeder Ausführung zuerst alle vorhandenen Links zwischen den Configuration Items, um die Zuordnung zwischen VM und ESX-Server aktuell zu halten. Eventuell manuell angelegte Links gehen dadurch verloren. Laut des vorgestellten Konzeptes, ist es allerdings auch nicht vorgesehen, dass Links zwischen CIs auf diese Weise angelegt werden.
Nach dem gleichen Prinzip können auch Importe von Links zwischen zwei Services und zwischen Geräten und Services realisiert werden. Die notwendigen Daten dafür müssten aber trotzdem manuell in einer CSV-Datei gepflegt werden, da eine automatische Erkennung dieser Strukturen nicht möglich ist. Hinzu kommt noch, dass man für die Objekte die entsprechende ID in der OTRS-Datenbank bestimmen müsste. Da dies wiederum sehr aufwändig wäre, ist es günstiger, solche Links direkt über das Interface von OTRS zu erzeugen und zu pflegen.

Das Shell Script, mit dem die Verlinkung erfolgt, befindet sich in Anlage 7. Für den Teil der darin enthaltenen Funktionen wäre es günstiger, wenn sie direkt im Framework von OTRS realisiert würden. Insbesondere der direkte Datenbankzugriff ist mit Risiken verbunden. So kann es zum Beispiel bei der Umstellung auf eine neue OTRS-Version auch zu Änderungen im Datenbankschema kommen, durch die im Script verwendeten SQL-Befehle ungültig werden. Hier würde die Nutzung einer abstrahierenden Schnittstelle sicherer sein. Eine auf einfachem Wege zugängliche Schnittstelle, wie die für den Import und Export von Configuration Items, existiert allerdings nicht. Im Rahmen der Arbeit ist es auch nicht möglich, soweit ins Framework einzusteigen und eine entsprechende Erweiterung oder Schnittstelle zu entwickeln. Die einzige Alternative wäre die Daten manuell über das Webinterface zu pflegen, was allerdings in der Praxis nicht realistisch ist. Die gleiche Problematik gilt auch für die Umsetzung der Nagios-Integration, die in Abschnitt 3.4 beschrieben wird.

3.3 Erstellung der Service-Struktur

Im Gegensatz zu den anderen beschriebenen Maßnahmen sind beim Erstellen und Pflegen der Service-Struktur vor allem manuelle Maßnahmen nötig. Datenbasis ist hier die Verfahrensdokumentation im internen IT-Wiki. Um einen neuen Service im OTRS zu erstellen, sind nur folgende Informationen nötig:

- Name
- Hierarchische Einordnung (Unterservice)
- Typ
- Kritikalität
- Gültigkeit
- Kommentar

Eine hierarchische Einordnung zur Kategorisierung ist nur notwendig, wenn man die Services zur Anzeige im Ticketbereich nach Relevanz sortieren will. So können mehrere Oberservices angelegt werden, denen die Services nach Bedeutung für den Service Desk untergeordnet werden. Alle sonstigen Abhängigkeiten werden hingegen über Verlinkungen dargestellt.
Für den Typ gibt es bereits eine vordefinierte Liste von Werten, über die man den Service in die Struktur des IT-Betriebs einordnen kann. Allerdings sind die Vorgaben für das Umfeld der SLUB nicht optimal, da es keine vollständig entsprechende Einordnung der Verfahren gibt. Es müssen daher eigene Typen angelegt werden. Eine sinnvolle Unterscheidung ist die von internen und externen Verfahren, also solchen, die eine Schnittstelle zu Anwendern in anderen Abteilungen oder außerhalb der SLUB beinhalten und solchen, bei denen dies nicht der Fall ist. Eine weitere Unterteilung ist zum Beispiel erst dann sinnvoll, wenn man gezielt die Verwendung von Services im OTRS auf bestimmte Benutzerkreise beschränken will. Der Typ kann nicht zu Kategorisierung der Services während der Ticketerstellung herangezogen werden, da er hier keine Auswirkungen hat.
Für die Kritikalität gibt es keine zuverlässigen Informationen, nach denen man die Verfahren einordnen kann. Theoretisch wäre hier die Verwendung der im Regelbetrieb der IT-Abteilung verwendeten Verfügbarkeitsklassen möglich. Allerdings ist diese nicht für alle Verfahren definiert. Daher wird sie, sofern sich in der Praxis nicht noch ein anderes Kriterium findet, für alle Services auf *normal* gesetzt.

Die Definition weiterer Felder für einen Service ist nicht möglich. Weitergehende Informationen zu einem Verfahren sind daher in FAQ-Einträgen bzw. wie bisher in der Dokumentation im internen Wiki zu finden.

Wenn alle Services definiert wurden, werden für sie entsprechend den Abhängigkeiten Verlinkungen erstellt. Wie schon oben erwähnt, können sowohl andere Services, als auch Configuration Items als Verlinkungsobjekt herangezogen werden. Als Verlinkungstyp stehen dabei verschiedene Möglichkeiten zur Auswahl, deren Verwendung unterschiedliche Bedeutung hat. Für die Verknüpfung der Services ist, wie auch im IT-Wiki vorgegeben, die Abhängigkeit zu wählen. Dadurch wird verdeutlicht, welcher Service den anderen benötigt. Ebenso wird für die Verknüpfung von Services mit verfahrenskritischen CIs verfahren. Alle CIs, die nur einen nebensächlichen Beitrag für den Service leisten, werden als relevant verknüpft. Ein wichtiger Aspekt muss vor dem Erstellen der Verknüpfungen beachtet werden. Im Normalzustand sieht OTRS keine Links zwischen zwei Services vor. Daher muss eine Ergänzung der Konfigurationsdateien vorgenommen werden [o]. Über ein XML-Schema wird diese Art der Verlinkung erlaubt und die gewünschten Verknüpfungstypen freigeschaltet.

3.4 Einbindung der Netzwerküberwachung

Um Nagios mit OTRS zu verbinden, gibt es bereits mehrere Ansätze. So ist es zum Beispiel möglich einige Parameter des Systems durch Nagios abzufragen und überwachen zu lassen [p]. Andersherum besteht die Möglichkeit von Nagios aus eine E-Mail-Benachrichtigung an OTRS zu senden, um daraus ein Ticket generieren zu lassen [q]. Diese Möglichkeit wurde auch schon in ersten Tests an der SLUB angewendet, befindet sich aber momentan nicht im Praxiseinsatz. Dies liegt vor allem daran, dass es noch kein geeignetes Konzept gibt, welches vorschreibt, wann eine Fehlermeldung im Nagios das Erstellen eines Tickets rechtfertigt und wann nicht. Wenn man alle Meldungen zulässt, führt dies zwangsläufig zu einer Überbelastung des Service Desks.
Dieses Problem muss man auch beachten, wenn man explizit an die Configuration Items Fehlermeldungen übermitteln will. Das Erstellen eines Tickets ist nicht praktikabel. Es muss daher direkt am Datenbestand des CIs gearbeitet werden. Am

besten ist dafür die Änderung des Wertes für den Vorfallsstatus geeignet. Zur Änderung dieses Wertes wäre es theoretisch denkbar eine neue Version des CIs zu erstellen, indem man einen weiteren Import des Gerätes auslöst. Dabei würde allerdings wieder die in Abschnitt 3.2 beschriebene Problematik des Speicherverbrauchs auftreten. Bei jeder Fehlermeldung würden wieder große Mengen redundanter Informationen in die Datenbank geschrieben werden. Wenn dann zum Beispiel bei Netzwerkproblemen viele Fehlermeldungen entstehen, wird die Datenbank sehr schnell überlastet. Die Konsequenz daraus ist, dass die Änderung direkt am Wert für den Vorfallsstatus in der Datenbank erfolgen muss.

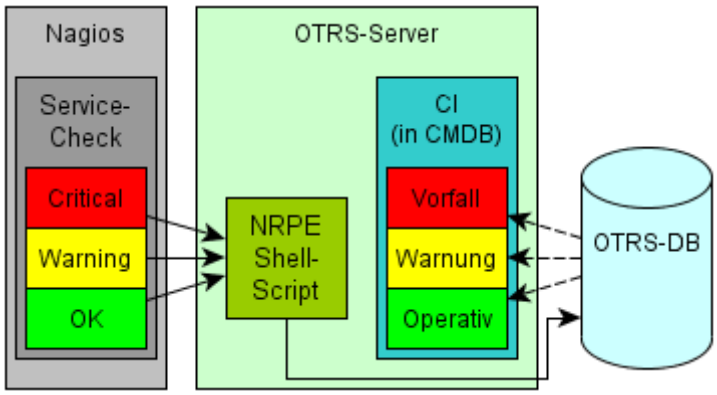

Abbildung 6. Einbindung von Überwachungsergebnissen aus Nagios

In Abbildung 6 wird das Grundprinzip der Übermittlung des Status aufgezeigt. Wenn Nagios für einen Service[3] eine Änderung ermittelt, wird ein Benachrichtigungsbefehl ausgeführt. Durch diesen wird, unter Verwendung des zugeordneten Hosts und des Status, per NRPE ein Shell Script auf dem OTRS-Server gestartet. Dieses ändert den Verwendungsstatus des entsprechenden Configuration Items in der OTRS-Datenbank durch Ausführung eines SQL-Befehls. Je nach Statusänderung ist dann für das CI im OTRS ein anderer Vorfallsstatus vorhanden. Die vordefinierten Möglichkeiten für den Vorfallsstatus im OTRS lassen sich ohne weitere Änderungen zur Abbildung des Status aus Nagios nutzen.

Die Ausführung über NRPE hat den Vorteil, dass dadurch vermieden wird, dass das OTRS-Datenbankpasswort auf einem anderen System hinterlegt wird. Sie ist außerdem notwendig, da Nagios nicht die Geräte in der CMDB identifizieren kann. Der Hostname lässt sich zwar im Befehl angeben, aber ist nur bei virtuellen

[3] Es ist der Service im Sinne von Nagios gemeint, also die regelmäßige Überwachung einer bestimmten Eigenschaft eines Hosts.

Maschinen als Identifikator nützlich. Auf dem OTRS-Server muss daher der Hostname auf die ID des CI gemappt werden. Dies geschieht dadurch, dass das Shell Script auf eine CSV-Datei zugreift, in der zeilenweise der Hostname aus Nagios und die Number des CIs aus der CMDB hinterlegt sind. Diese wird per automatischen Export aus OTRS erzeugt. Das Prinzip dabei ist das gleiche wie beim Import. Es wird pro Klasse ein Export-Template angelegt, welches über die Import-Export-Schnittstelle aufgerufen wird. Die erhaltenen CSV-Dateien werden zur einfacheren Weiterverarbeitung zusammengefasst. Die Erstellung der CSV-Dateien erfolgt im Zuge des täglichen Datenimports.
Für den Status muss ebenfalls ein Mapping erfolgen. Der Status wird von Nagios als String ausgegeben und muss durch die entsprechende Vorfallsstatus-ID im OTRS ersetzt werden. Welcher Status auf welchen Vorfall gemappt wird, lässt sich aus der farblichen Darstellung in Abbildung 6 erkennen.
In Nagios muss die Ausführung des NRPE-Befehls als Benachrichtigungsbefehl definiert werden. Dies geschieht an der gleichen Stelle, an der auch die E-Mail-Benachrichtigung eingestellt ist. In den Systemen der SLUB ist als Befehlszeile der folgende Wert einzutragen:

```
$USER1$/check_nrpe -H otrsserver -c updatestatus -a $HOSTNAME$ $SERVICESTATE$
```

Durch diesen Aufruf wird auf dem Host *otrsserver* der Befehl *updatestatus* mit den Werten für den Hostnamen und den Servicestatus als Parameter durchgeführt. Der Befehl muss zusätzlich in der NRPE-Konfiguration des OTRS-Servers definiert werden. Dies geschieht wie folgt:

command[updatestatus]=/opt/scripts/nagios.sh $ARG1$ $ARG2$

Das letztendlich damit ausgeführte Shell Script findet sich in Anlage 8.

Nach der Definition der Befehle muss noch geklärt werden, wodurch sie im Nagios ausgelöst werden. Die aktuelle Definition ist nur darauf ausgerichtet, Statusänderungen bei Services zu erkennen. Dementsprechend ist sie auch nur als Benachrichtigungsbefehl bei Service-Checks möglich und nicht bei Host-Checks.

Eine Beschränkung auf Host-Checks wäre zudem auch nicht ausreichend, da hier nur per Ping geprüft wird. Fehler bei einem Service würden so nicht erkannt werden.
Ein Problem in der aktuellen Konfiguration ist, dass entweder nur die E-Mail-Benachrichtigung oder nur die OTRS-Status-Änderung ausgeführt werden kann. Um beides zu ermöglichen, muss ein Shell-Script auf dem Nagios-Server erstellt werden, das beide Aufrufe ausführt. Der Aufruf des Shell-Scripts ist dann als Alternative zu den Einzelaufrufen in der Nagios-Konfiguration zu erstellen. Ein Konzept, wann welche Benachrichtigungsform gewählt wird, muss noch erstellt werden.

4 Verwendung der neuen Funktionen

Die Mitarbeiter des Service Desks der SLUB erhalten durch die Umsetzung des beschriebenen Konzepts einen einfachen und zentralen Zugang zu relevanten Informationen - für den Preis eines etwas höheren Pflegeaufwands. Dieser Aufwand entsteht allerdings primär beim ersten Anlegen der vorhandenen Strukturen.

Abgesehen von dem Zugang zur CMDB und zu Informationen über Services, ändert sich für die Agenten im OTRS erst einmal nicht viel. Auffälligste Änderung ist das freigeschaltete Feld zur Auswahl des Services. Sofern sich die Meldung des Kunden auf ein Verfahren der IT zurückführen lässt, ist dieses bei der Erstellung bzw. im Laufe der Bearbeitung zu belegen. Dadurch dient es später als schneller Hinweis für andere Agenten, welches Verfahren betroffen ist. Zudem steht es dann auch für statistische Auswertungen zur Verfügung.
Inwieweit die Agenten in die Pflege der FAQs und die Verknüpfung von Tickets mit Configuration Items involviert werden, muss noch festgelegt werden. Dies hängt vor allem davon ab, welche weitergehenden Veränderungen im Workflow auf Grundlage der vorgenommenen Änderungen realisiert werden sollen.
Das Pflegen der beschriebenen Verknüpfungsstrukturen ist Aufgabe der Agenten, die das jeweilige Verfahren betreuen bzw. Informationen zu bekannten Fehlern liefern können. Festlegungen dafür existieren allerdings noch nicht und müssen auch noch definiert werden.

Für die Administratoren besteht der Hauptaufwand in der Erstellung der entsprechenden Strukturen in der OTRS-Konfiguration. Insbesondere im General Catalog müssen bei der ersten Konfiguration zahlreiche Anpassungen und Ergänzungen vorgenommen werden. Die Scripte müssen in den Bereichen, bei denen noch direkt auf der Datenbank gearbeitet wird, an das entsprechende System angepasst werden. Die weitere Ausführung der Scripte ist hingegen automatisiert. Wenn die Strukturen erstellt sind, werden sie automatisch mit Daten gefüllt und auf einem aktuellen Stand gehalten. Änderungen in den Scripten müssen dann nur noch erfolgen, wenn neue Geräteklassen im BVZ angelegt oder die Attribute bestehenden Klassen geändert werden. Bei Nutzung neuer ESX-Server muss das Script zur Verlinkung von VM und ESX erweitert werden. Auch bei Nutzung einer neuen Version von OTRS ist die Notwendigkeit von Änderungen nicht ausgeschlossen. Generell sind die Scripte gut durch Kommentare beschrieben. Eine allgemeine Dokumentation der umgesetzten Schritte (außerhalb dieser Arbeit) wird noch vorgenommen. Damit sollten Änderungen auch durch andere Administratoren problemlos möglich sein.
Die bestehenden Scripte werden für die Umsetzung im Produktivsystem noch optimiert und in einem einfach zu installierendem Softwarepaket zusammengefasst. Auf dem zweiten OTRS-Server, das primär als Testumgebung dient, ist die Nutzung der Scripte nicht notwendig. Hier reicht der seit Längerem umgesetzte Datenabgleich mit dem Primärsystem aus. Die weitere Nutzung auf anderen Testsystemen ist ebenfalls möglich und liegt im Ermessen des verantwortlichen Mitarbeiters.

5 Erfolgsbetrachtung und Ausblick

Das Hauptziel der Arbeit war die Schaffung einer technischen Basis im OTRS über die eine CMDB ins System integriert wird. Dieses Ziel wurde unter Berücksichtigung der Anforderungen von ITIL und der SLUB erreicht. Die Machbarkeit wurde durch die praktische Umsetzung der wesentlichen Bestandteile in einem Testsystem nachgewiesen und ist daher auch für die Umsetzung im Produktivsystem geeignet.
Es sind allerdings auch noch Verbesserungsmöglichkeiten bei diversen Teilkomponenten gegeben. Es ist möglich, die umgesetzten Funktionen stärker für das Framework von OTRS auszurichten, um den direkten Datenbankzugriff zu vermeiden. Auch in der Gestaltung der Scripte können an einigen Stellen der Pflege-

und der Rechenaufwand optimiert werden. Hauptproblem bei der Umsetzung war die Erstellung neuer Versionen von Configuration Items durch OTRS beim Import. Da überflüssige Versionen vermieden werden mussten, gestaltete sich der Datenabgleich sehr aufwendig. Hier wäre es günstiger, wenn die Import-Export-Schnittstelle des OTRS erkennen würde, ob sich die Daten eines CIs geändert hätten und dementsprechend nur in diesem Fall einen Import erlaubt. Zudem ist auch eine Schnittstelle zum automatischen Import von Verknüpfung wünschenswert.

Die Nutzungsmöglichkeiten, der durch die Änderungen hinzugekommenen Daten, wurden in einem Rahmen der direkten Anwendbarkeit in OTRS analysiert. Auf ihre Verwendungsmöglichkeiten bei der Erweiterung oder Umgestaltung von Workflows wurde hingegen nur hingewiesen. Hier müssen weiterführende Konzepte geschaffen werden, über die eine praktische Verbesserung der Arbeit des Service Desks erzielt wird. Insbesondere die Pflege einer KEDB, eines Configuration Models oder eine weitergehende Nutzung von Services und SLAs sind Punkte, an denen man ansetzen kann.

Langfristig gesehen können die Änderungen, insbesondere die Daten in der CMDB, auch als Basis zur Verwaltung aller gerätebezogenen Incidents dienen und somit das BVZ in dieser Funktion ablösen. Allerdings wären dafür zahlreiche Änderungen im Interface von OTRS notwendig. Vor allem der Workflow zur Verknüpfung von Tickets und CIs ist für den Praxiseinsatz nicht unbedingt geeignet. Hier wäre es wünschenswert, wenn bereits beim Prozess der Ticketerstellung das entsprechende CI gewählt werden könnte. In der aktuellen Umsetzung von OTRS ist dies nur danach möglich und ist auch nicht sehr anwenderfreundlich gestaltet.
Sollte es gelingen den Workflow durch Anpassungen in diesem Bereich entsprechend den Anforderungen der SLUB zu optimieren, ist es auch denkbar, die Bestandsverwaltung komplett ins OTRS zu verlegen. Sämtliche gerätebezogenen Vorgänge würden dann in Tickets verwaltet und in festgelegten Workflows abgearbeitet werden. In diesem Fall müsste man den in der Arbeit geschaffenen automatischen Importprozess stoppen und die letzte damit erstellte Version der CMDB als Basis für alle zukünftigen Änderungen verwenden.

6 Quellen- und Literaturverzeichnis

[1] CANNON, David ; WHEELDON, David : Service Operation : Stationery Office Books, 2007, Seite 198 ff.

[2] CANNON, David ; WHEELDON, David : Service Operation : Stationery Office Books, 2007, Seite 216 f.

[3] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 68 f.

[4] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 67.

[5] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 75.

[6] CANNON, David ; WHEELDON, David : Service Operation : Stationery Office Books, 2007, Seite 123 f.

[7] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 65.

[8] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 66 f.

[9] LACY, Shirley ; MACFARLANE, Ivor : Service Transition : Stationery Office Books, 2007, Seite 72 f.

7 Verzeichnis der Onlinequellen

[a] What is ITIL, APM Group, (Stand: 24. Juni 2009)
http://www.itil-officialsite.com/AboutITIL/WhatisITIL.asp [Online]

[b] ITIL V3 Glossar, itSMF, (Stand: 25. Juni 2009)
http://www.itsmf.de/fileadmin/dokumente/AK_Publikationen/20070831_ITIL_V3_Glossary_Germany.pdf [Online]

[c] OTRS::ITSM - das OTRS für IT Service Management, OTRS Dokumentation (Deutsch), (Stand: 17. Juli 2009)
http://doc.otrs.org/itsm/1.2/de/html/c29.html [Online]

[d] OTRS::ITSM, OTRS AG, (Stand: 17. Juli 2009)
http://www.otrs.com/de/produkte/otrsitsm/ [Online]

[e] Anwenderhandbuch für OTRS::CiCS 2.1, c.a.p.e. IT, (Stand: 17. Juli 2009)
http://www.cape-it.de/cgi-bin/download.pl/OTRS_CiCS_Manual_2.1.pdf [Online]

[f] Konfiguration der Configuration Item Klassen, OTRS Dokumentation (Deutsch), (Stand: 9. Juli 2009)
http://doc.otrs.org/itsm/1.2/de/html/x714.html [Online]

[g] Optimize Table Syntax, MySQL 5.1 Reference Manual, (Stand: 7. Juli 2009)
http://dev.mysql.com/doc/refman/5.1/en/optimize-table.html [Online]

[h] Mit Perl auf Datenbanken zugreifen, Teil 5, freeX, (Stand: 2. Juli 2009)
http://www.infodrom.org/~joey/Writing/freeX/perl5/ [Online]

[i] CASE (Transact-SQL), Microsoft Developer Network, (Stand: 2. Juli 2009)
http://msdn.microsoft.com/de-de/library/ms181765.aspx [Online]

[j] Pivot-Tabellen mit dem SQL Server 7.0 erzeugen, itrain, (Stand: 2. Juli 2009)
http://www.itrain.de/knowhow/sql/tsql/pivot/pivot.asp [Online]

[k] Date/Time Functions and Operators, PostgreSQL, (Stand: 20. Juli 2009)
http://www.postgresql.org/docs/8.1/static/functions-datetime.html [Online]

[l] Using the vmware-cmd Utility, VMware, (Stand: 6. Juli 2009)
http://www.vmware.com/support/esx21/doc/vmware-cmd.html [Online]

[m] Whats in the VirtualCenter database?, VMware Communities, (Stand: 6. Juli 2009)
http://communities.vmware.com/thread/81313 [Online]

[n] How to connect from perl to MS SQL, My Read the Fine Manual page, (Stand: 6. Juli 2009)
http://myrtfm.blogspot.com/2008/09/how-to-connec-from-perl-to-ms-sql.html [Online]

[o] CI Klassen & Services für Rechenzentren in OTRS::ITSM, Jens Bothe, (Stand: 11. Juli 2009)
http://www.netways.de/uploads/media/Jens_Bothe_OTRS_ITSM.pdf [Online]

[p] OTRS - Nagios Integration, Jens Bothe, (Stand: 12. Juli 2009)
http://www.netways.de/uploads/media/
Jens_Bothe_Nagios___OTRS_Integration_02.pdf [Online]

[q] Integration von Nagios und OTRS, Wolfgang Barth, (Stand: 12. Juli 2009)
http://www.netways.de/uploads/media/
Wolfgang_Barth_Integration_von_Nagios_und_OTRS_02.pdf [Online]

Alle URLs wurden zuletzt am 29.07.2009 auf ihre Aktualität geprüft.

8 Verzeichnisse der Abbildungen, Tabellen, Anlagen

Abbildungen

Tabellen

Anlagen

Übersicht zu Geräteklassen aus dem BVZ
Anlage 1

Klasse	Id		**Status**		**Allg. Daten**				**Verantwort.**		**Attribute**						Sonstiges
Computer	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Prozessor-typ	Taktrate	Anzahl HDD	Platz-nummer	IP	Person	Bemerkungen
Drucker	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Farbe	Papier	Typ	Platz-nummer	IP	Person	Bemerkungen
Netz-drucker	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Typ	Farbe	Papier	Platz-nummer	IP	Person	Bemerkungen
Barcode-scanner	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort							Bemerkungen
Projektor/ Beamer	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Typ						Bemerkungen
Monitor	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Größe						Bemerkungen
Alt-bestand	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort							Bemerkungen
Extern/ Zubehör	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Brenner-typ	Typ					Bemerkungen
Print-server	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Platz nummer	IP	Person				Bemerkungen
Scanner	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Typ	Größe					Bemerkungen
Notebook	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Taktrate	Platz-nummer	IP	Person			Bemerkungen
Docking-station	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort							Bemerkungen
Netzkom-ponente	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Kompo-nente						Bemerkungen
USV	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Leistung						Bemerkungen
Zubehör	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort							Bemerkungen
Selbst-bedienung	ID	Name	Verwen-dungsstatus	Vorfalls-status	Inventarnr.	Seriennr.	Datum Lieferung	Datum Garantieablauf	Name	Standort	Typ	Platz-nummer	IP	Person			Bemerkungen

Werte

aus BVZ
aus Script

Definition der CI-Klasse Computer
Anlage 2

```
[
    {
        Key => 'Inventarnummer',
        Name => 'Inventarnummer',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Seriennummer',
        Name => 'Seriennummer',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Lieferdatum',
        Name => 'Lieferdatum',
        Searchable => 1,
        Input => {
            Type => 'Date',
        },
    },
    {
        Key => 'Garantieablauf',
        Name => 'Garantieablauf',
        Searchable => 1,
        Input => {
            Type => 'Date',
        },
    },
    {
        Key => 'Verantwortlicher',
        Name => 'Verantwortlicher',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Standort',
        Name => 'Standort',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Prozessortyp',
        Name => 'Prozessortyp',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
        Sub => [
            {
                Key => 'Taktrate',
                Name => 'Taktrate',
                Input => {
                    Type => 'Text',
                    Size => 10,
                    MaxLength => 10,
                },
            },
        ],
    },
{
        Key => 'Festplatten',
        Name => 'Festplatten',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 3,
            MaxLength => 3,
        },
    },
    {
        Key => 'Platznummer',
        Name => 'Platznummer',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'IP',
        Name => 'IP',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Anwender',
        Name => 'Anwender',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
    {
        Key => 'Bemerkung',
        Name => 'Bemerkung',
        Searchable => 1,
        Input => {
            Type => 'Text',
            Size => 50,
            MaxLength => 50,
        },
    },
];
```

Shell-Script zum Import der Daten von physischen Geräten
Anlage 3

```
#!/bin/bash

#Aufruf Perl-Script BVZ-Abfrage
perl -X computer_export.pl
printf  "Export abgeschlossen\n\n"

#Datenmanipulation: Einfügen des Standard-Vorfallsstatus |
Zusammenfassen von Vor- und Nachname | Ersetzen der
Bezeichnung für fehlende Geräteattribute | Löschen von
Einträgen ohne Inventarnummer
sed 's/^\([^;]*";"[^;]*";"[^;]*";"\)/\1Operational";"/g'
export.csv | sed
's/^\([^;]*;[^;]*;[^;]*;[^;]*;[^;]*;[^;]*;[^;]*;[^;]*;[^;]*\)"
;"/\1, /' | sed 's/zzz/Keine Angabe/g' | sed '/"@ohne"/d'
>result.csv
rm export.csv
printf "Datenanpassung abgeschlossen\n\n"

#Aufruf Perl-Script CI-Import
/opt/otrs/bin/ImportExport.pl -n 3 -a import -i ./result.csv
rm result.csv
printf "\n"

#Optimierung der Datenbankeinträge
echo "use otrs4; optimize table xml_storage;" | mysql -h *** -
u *** -p***
printf "\nGesamtvorgang abgeschlossen\n"
```

Perl-Script zur Abfrage der BVZ-Datenbank
Anlage 4

```
#!/usr/bin/perl -w
# --
# --

   use DBI;

   $dsn = "dbi:Pg:dbname=***;host=***;port=5432";
   $dbh = DBI->connect($dsn, "***", "***");
   if (!$dbh) {
       print "Zugriff verweigert!\n";
       exit (1);
   }

   $query  = "<siehe Anlage 5>";

   $sth = $dbh->prepare($query);

   $rv = $sth->execute;

   if ($rv <= 0) {
       print "Keine Ergebniswerte!\n";
       exit (1);
   }

   open(DATEI1,">export.csv") || die "Fehler beim Erstellen
der Datei";

   while (@row = $sth->fetchrow_array) {
     printf DATEI1
"\"".$row[0]."\";\"".$row[1]."\";\"".$row[2]."\";\"".$row[3]."
\";\"".$row[4]."\";\"".$row[5]."\";\"".$row[6]."\";\"".$row[7]
."\";\"".$row[8]."\";\"".$row[9]."\";\"".$row[10]."\";\"".$row
[11]."\";\"".$row[12]."\";\"".$row[13]."\";\"".$row[14]."\";\"
".$row[15]."\"\n";
   }

   close(DATEI1);

   $dbh->disconnect if ($dbh);
```

SQL-Abfrage für die BVZ-Datenbank
Anlage 5 – Seite 1

```
SELECT topq_1.dsn, topq_1.bezeichnung, topq_3.status, topq_1.inv, topq_1.sn, topq_1.d_lief, topq_1.d_gara,
topq_1.nachname, topq_1.vorname, topq_1.bezeichung AS location, topq_2.Prozessortyp, topq_2.Taktrate, topq_2.AnzahlHDD,
topq_1.platznr, topq_1.ip, topq_1.person, topq_1.bemerkung
FROM

      (SELECT gn.dsn, gn.dsn_klasse, gn.inv, gn.sn, gn.bezeichnung, gn.d_lief, gn.d_gara, subq_1.nachname,
subq_1.vorname, subq_1.bezeichung, gn.platznr, gn.ip, gn.person, gn.bemerkung
      FROM gn INNER JOIN
            (SELECT ma_ap_location.dsn, verantw.nachname, verantw.vorname, location.bezeichung
            FROM ma_ap_location
            INNER JOIN verantw ON ma_ap_location.dsn_ma=verantw.dsn
            INNER JOIN location ON ma_ap_location.dsn_location=location.dsn
            ) AS subq_1
            ON gn.dsn_verantw=subq_1.dsn
      WHERE dsn_klasse=1
      ) AS topq_1

INNER JOIN

      (SELECT subq_2.dsn_gn,
            MIN(CASE subq_2.eigenschaft WHEN 'Anschluss' Then subq_2.wert ELSE 'zzz' END) As Anschluss,
            MIN(CASE subq_2.eigenschaft WHEN 'Größe' Then subq_2.wert ELSE 'zzz' END) As Groesse,
            MIN(CASE subq_2.eigenschaft WHEN 'Technologie' Then subq_2.wert ELSE 'zzz' END) As Technologie,
            MIN(CASE subq_2.eigenschaft WHEN 'Anzahl HDD' Then subq_2.wert ELSE 'zzz' END) As AnzahlHDD,
            MIN(CASE subq_2.eigenschaft WHEN 'Anzahl Prozessoren' Then subq_2.wert ELSE 'zzz' END) As AnzahlProzessoren,
            MIN(CASE subq_2.eigenschaft WHEN 'Brennertyp' Then subq_2.wert ELSE 'zzz' END) As Brennertyp,
            MIN(CASE subq_2.eigenschaft WHEN 'Datennetzzugang' Then subq_2.wert ELSE 'zzz' END) As Datennetzzugang,
            MIN(CASE subq_2.eigenschaft WHEN 'Farbe' Then subq_2.wert ELSE 'zzz' END) As Farbe,
            MIN(CASE subq_2.eigenschaft WHEN 'Komponente' Then subq_2.wert ELSE 'zzz' END) As Komponente,
            MIN(CASE subq_2.eigenschaft WHEN 'Leistung' Then subq_2.wert ELSE 'zzz' END) As Leistung,
            MIN(CASE subq_2.eigenschaft WHEN 'Papier' Then subq_2.wert ELSE 'zzz' END) As Papier,
            MIN(CASE subq_2.eigenschaft WHEN 'Prozessortyp' Then subq_2.wert ELSE 'zzz' END) As Prozessortyp,
            MIN(CASE subq_2.eigenschaft WHEN 'Taktrate' Then subq_2.wert ELSE 'zzz' END) As Taktrate,
            MIN(CASE subq_2.eigenschaft WHEN 'Typ' Then subq_2.wert ELSE 'zzz' END) As Typ,
            MIN(CASE subq_2.eigenschaft WHEN 'Zubehör' Then subq_2.wert ELSE 'zzz' END) As Zubehoer
      FROM
```

SQL-Abfrage für die BVZ-Datenbank
Anlage 5 – Seite 2

```
            (SELECT gn_wert.dsn_gn, subq_7.eigenschaft, subq_7.wert
            FROM gn_wert INNER JOIN
                  (SELECT wert.dsn, wert.wert, wert.dsn_eigenschaft, eigenschaft.eigenschaft
                  FROM wert INNER JOIN eigenschaft ON wert.dsn_eigenschaft=eigenschaft.dsn
                  ) AS subq_7
            ON gn_wert.dsn_wert=subq_7.dsn
            ) AS subq_2
      GROUP BY subq_2.dsn_gn
      ORDER BY dsn_gn
      ) AS topq_2

ON topq_1.dsn=topq_2.dsn_gn
INNER JOIN

      (SELECT subq_4.dsn_gn, subq_5.status
      FROM
            (SELECT statuschange.dsn_gn,  max(statuschange.datum) As MaxDatum
            FROM statuschange
            GROUP BY statuschange.dsn_gn
            ORDER BY statuschange.dsn_gn
            ) AS subq_4
      INNER JOIN
            (SELECT statuschange.dsn_gn, statuschange.datum, statuschange.dsn_status_new, status.status
            FROM statuschange INNER JOIN status ON statuschange.dsn_status_new=status.dsn
            ORDER BY statuschange.dsn_gn
            ) AS subq_5
      ON subq_4.MaxDatum=subq_5.datum AND subq_4.dsn_gn=subq_5.dsn_gn
      WHERE subq_5.datum > now() - interval '1 days'
      ) AS topq_3

ON topq_1.dsn=topq_3.dsn_gn
ORDER BY dsn
```

Shell-Script zum Import der Daten von virtuellen Maschinen
Anlage 6

```
#!/bin/bash

#Aufruf Perl-Script Virtual Center Datenbank-Abfrage
#todo

#Datenmanipulation: Löschen der Header-Zeile | Löschen von Leerzeilen  |
Löschen von Kommentaren mit Kommas | Ersetzen von Kommas mit Semikolon |
Einfügen des Standard-Vorfallsstatus | Ändern des State (2 mal) | Kürzung
des Hostname
sed '/NAME[A-Z, ]*NOTES/d' esx.csv | sed '/^\s*$/d' | sed 's/".*ß\?.*"//g'
| sed 's/,/;/g' | sed 's/^\([^;]*;[^;]*;\)/\1Operational;/g' | sed
's/Powered On/im Einsatz/g' | sed 's/Powered Off/inventarisiert/g' | sed
's/.slub-dresden.de//g' >esx2.csv
sleep 1

printf "\nDatenanpassung abgeschlossen\n\n"

#Export vorhandener VMs zum Vergleich, Zeichensatzanpassung, Entfernen von
Anführungszeichen, Einfügen des letzten Zeilenumbruchs
/opt/otrs/bin/ImportExport.pl -n 4 -a export -o esx_export.csv
sed 's/";"/;/g' esx_export.csv | sed 's/^"//g' | sed 's/"$//g'
>esx_export2.csv
printf "\n" >>esx_export2.csv
printf "\nExport und Anpassung der Vergleichsdaten abgeschlossen\n\n"

#Vergleich und Gruppierung geänderter Daten
sort -o tmpfile2.csv esx2.csv
sort -o tmpfile1.csv esx_export2.csv
comm -3 --output-delimiter=XXXXX  tmpfile1 tmpfile2 >importfile.csv
cat importfile.csv | grep XXXXX | sed 's/XXXXX//' >import1.csv
cat importfile.csv | grep -v XXXXX | sed 's/im Einsatz/ausgesondert/g' |
sed 's/im Einsatz/ausgesondert/g' >import2.csv

cp import2.csv import3.csv
for LINE in `sed 's/^\([^;]*\).*/\1/' import1.csv`
do
      sed /^$LINE,/d import2.csv >import3.csv
      cp import3.csv import2.csv
done


#Aufruf Perl-Script CI-Import
/opt/otrs/bin/ImportExport.pl -n 4 -a import -i ./import1.csv
/opt/otrs/bin/ImportExport.pl -n 4 -a import -i ./import2.csv

#Alle CSVs außer Originaldaten löschen
mv esx.csv esx.bak
rm *.csv
mv esx.bak esx.csv

printf "\nImport beendet\n\n"

#Optimierung der Datenbankeinträge
echo "use otrs4; OPTIMIZE TABLE xml_storage;" | mysql -h *** -u *** -p***
printf "\nGesamtvorgang abgeschlossen\n"
```

Shell-Script zur Verlinkung von VMs und ESX-Servern
Anlage 7

```
#!/bin/bash

# Erzeugung der CSV-Datei mit Geräte-IDs
sed '/NAME[A-Z, ]*NOTES/d' esx.csv | sed '/^\s*$/d' | sed 's/".*ß\?.*"//g'
| sed 's/,/;/g' | sed 's/^\([^;]*;[^;]*;\)/\1Operational;/g' | sed
's/Powered On/im Einsatz/g' | sed 's/Powered Off/inventarisiert/g' | sed
's/.slub-dresden.de//g' >esx2.csv
iconv -f ISO-8859-1 -t UTF-8 esx2.csv | sed
's/^\([^;]*\);[^;]*;[^;]*;[^;]*;[^;]*;\(sdvvmesx[01]\?[0-9]\);.*/\2;\1/g'
>esx_links.csv
sed '/sdvvmesx01/d' esx_links.csv | sed '/sdvvmesx02/d' | sed
'/sdvvmesx03/d' | sed 's/sdvvmesx04/985/g' | sed 's/sdvvmesx05/987/g' | sed
's/sdvvmesx06/986/g' | sed 's/sdvvmesx07/988/g' | sed 's/sdvvmesx08/989/g'
| sed 's/sdvvmesx09/1004/g' | sed 's/sdvvmesx10/1005/g' | sed
's/sdvvmesx11/1008/g' | sed 's/sdvvmesx12/1007/g' | sed
's/sdvvmesx13/1006/g' | sed 's/sdvvmesx14/1011/g' | sed
's/sdvvmesx15/1010/g' | sed 's/sdvvmesx16/1009/g' | sed '/^\s$/d'
>esx_links2.csv

printf "\nDatenanpassung abgeschlossen\n\n"

# Bestimmung der Geräte-IDs
count=0
for line in `sed 's/\([^;]*\);\(.*\).*/\2/g' esx_links2.csv`
do
  ident[$count]=`echo "use otrs4; select distinct configitem_id from
configitem_version where name like '$line';" | mysql -h *** -u *** -p*** |
grep [0-9]`
  count=`expr $count + 1`
done

printf "\nIDs ausgelesen\n\n"

objectid=`echo "use otrs4; select id from link_object where name like
'ITSMConfigItem';" | mysql -h *** -u *** -p*** | grep [0-9]`

echo "use otrs4; DELETE FROM link_relation WHERE source_object_id=$objectid
AND target_object_id=$objectid" | mysql -h *** -u *** -p***

linktype=`echo "use otrs4; select id from link_type where name like
'ConnectedTo';" | mysql -h *** -u *** -p*** | grep [0-9]`

# Erzeugung der Links
count=0
for line in `sed 's/\([^;]*\);\([^;]*\)/\1/g' esx_links2.csv`
do
  echo "use otrs4; INSERT INTO link_relation VALUES
('$objectid','${ident[$count]}','$objectid','$line','$linktype',1,NOW(),1);
" | mysql -h *** -u *** -p***
  count=`expr $count + 1`
done

printf "\nLinks erzeugt\n\n"

rm esx2.csv
rm esx_links.csv
rm esx_links2.csv
```

Shell-Script zur Änderung des Vorfallsstatus über NRPE
Anlage 8

```
#! /bin/sh

HOST=$1
FEHLER=$2

# Ermittlung der ID für den Vorfallsstatus
if [ $FEHLER = "CRITICAL" -o $FEHLER = "UNKNOWN" ];
      then ERROR=`echo "use otrs4; SELECT id FROM general_catalog WHERE
name LIKE 'Incident';" | mysql -h *** -u *** -p*** | grep [0-9]`
fi
if [ $FEHLER = "WARNING" ];
      then ERROR=`echo "use otrs4; SELECT id FROM general_catalog WHERE
name LIKE 'Warning';" | mysql -h *** -u *** -p*** | grep [0-9]`
fi
if [ $FEHLER = "OK" ];
      then ERROR=`echo "use otrs4; SELECT id FROM general_catalog WHERE
name LIKE 'Operational';" | mysql -h *** -u *** -p*** | grep [0-9]`
fi

#Number aus exportierter CI-Daten ermitteln
NUMBER=0
NUMBER=`cat /opt/scripts/all_export.csv | grep -i $HOST | sed
's/^"\([^;]*\)";"[^;]*"/\1/g'`

#Update des Vorfallsstatus
if [ -n "$NUMBER" ];
      then echo "use otrs4; UPDATE configitem SET cur_inci_state_id=$ERROR
WHERE configitem_number=$NUMBER;" | mysql -h *** -u otrs4 -p***
fi
echo "Beendet"
exit 0
```

Beschreibungen der verwendeten Systeme
Anlage 8

- **OTRS:**
  Das OTRS ist ein Ticketsystem, das zur Unterstützung des Service Desks eingesetzt wird. Es handelt sich um ein lizenzkostenfreies Open-Source-Projekt und läuft an der SLUB auf einem Debian-Linux-System.
  An der SLUB wird OTRS primär dazu genutzt verschiedene Anfragen von Anwendern zentral von der Annahme bis zum Abschluss zu verwalten. Zurzeit ist OTRS in der Version 2.2.7 im Einsatz und ist mit der Oberflächenmodifikation CiCS erweitert.

- **ITSM:**
  Bei ITSM handelt es sich um ein Paket für OTRS, das zahlreiche Funktionen für das Service Management bietet. Es dient unter anderem als Basis zur Umsetzung einer CMDB.

- **BVZ:**
  Das BVZ ist eine Eigenentwicklung der SLUB, die auf die bestehenden individuellen Anforderungen zugeschnitten wurde. Es dient vorwiegend zur Verwaltung von Bestandsdaten aller physischen Geräte im IT-Bereich, aber auch zur Erfassung von Störungen.

- **Nagios:**
  Nagios bietet eine Plattform, über die verschiedene Aspekte von Netzwerken und den daran angebundenen Geräten überwacht und ausgewertet können. Die Abfragen werden dabei über Plugins durchgeführt und sind daher individuell an verschiedenste Anforderungen anpassbar.

- **NRPE:**
  Durch NRPE wird Nagios ermöglicht Plugins direkt auf anderen Systemen auszuführen. Im Wesentlichen wird dabei durch NRPE eine Schnittstelle geschaffen an die Nagios Befehle senden kann. Diese Befehle werden dann durch NRPE interpretiert und ausgeführt. Ein Übergabe von Parametern ist mit vorgesehen.